Assume-Guarantee Synthesis for Digital Contract Signing

Krishnendu Chatterjee[1] and Vishwanath Raman[2]

[1] IST Austria (Institute of Science and Technology Austria)
krishnendu.chatterjee@ist.ac.at
[2] Carnegie Mellon University, Moffett Field, USA
vishwa.raman@sv.cmu.edu

Abstract. We study the automatic synthesis of fair non-repudiation protocols, a class of fair exchange protocols, used for digital contract signing. First, we show how to specify the objectives of the participating agents and the trusted third party (TTP) as path formulas in LTL and prove that the satisfaction of these objectives imply *fairness*; a property required of fair exchange protocols. We then show that *weak (co-operative) co-synthesis* and *classical (strictly competitive) co-synthesis* fail, whereas *assume-guarantee synthesis (AGS)* succeeds. We demonstrate the success of assume-guarantee synthesis as follows: (a) any solution of assume-guarantee synthesis is *attack-free*; no subset of participants can violate the objectives of the other participants; (b) the Asokan-Shoup-Waidner (ASW) certified mail protocol that has known vulnerabilities is not a solution of AGS; (c) the Kremer-Markowitch (KM) non-repudiation protocol is a solution of AGS; and (d) AGS presents a new and symmetric fair non-repudiation protocol that is attack-free. To our knowledge this is the first application of synthesis to fair non-repudiation protocols, and our results show how synthesis can both automatically discover vulnerabilities in protocols and generate correct protocols. The solution to assume-guarantee synthesis can be computed efficiently as the secure equilibrium solution of three-player graph games.

1 Introduction

Digital contract signing. The traditional paper-based contract signing mechanism involves two participants with an intent to sign a piece of contractual text, that is typically in front of them. In this case, either both of them agree and sign the contract or they do not. The mechanism is "fair" to both participants in that it does not afford either participant an unfair "advantage" over the other. In digital contract signing, ubiquitous in the internet era, an *originator* sends her intent to sign a contractual text to a *recipient*. Over the course of a set of messages, they then proceed to exchange their actual signatures on the contract. In this case, it is in general difficult to ensure fairness as one of the two participants gains an advantage over the other, during the course of the exchange. If the participants do not trust each other, then neither wants to sign the contract first and the one that signs it first may never get a reciprocal signature from the other participant. Moreover, as these contracts are typically signed over asynchronous networks, the communication channels may provide no guarantees on message delivery. The same situation arises in other related areas, such as fair exchange and certified email.

Protocols for digital contract signing. Many protocols have been designed to facilitate the exchange of digital signatures. The earliest exchange protocols were probabilistic. Participants transmit successive bits of information, under the expectation that both participants have similar computation power to detect dishonest behavior and stop participating in the

protocol. These protocols are impractical as the number of messages exchanged may be very large, and both participants having similar computation power may not be realistic. Even and Yacobi [12] first showed that no deterministic contract signing protocol can be realized without the involvement of a third party arbitrator who is trusted by all participants. This was formalized as an impossibility result in [21], where the authors show that fair exchange is impossible without a *trusted third party (TTP)* for non-repudiation protocols. A simple protocol with a TTP has a TTP collect all signatures and then distribute them to the participants. But this is inefficient as it involves an online TTP to facilitate every exchange, easily creating a bottleneck at the site of the TTP. This has lead to the development of *optimistic protocols*, where two participants exchange their signatures without involving a TTP, calling upon the TTP to adjudicate only when one of the two participants is dishonest. These protocols are called *fair non-repudiation protocols* with *offline* TTP.

Fair non-repudiation protocols. A *fair non-repudiation* protocol is therefore a contract signing protocol, falling under the category of fair exchange protocols, that ensures that at the end of the exchange of signatures over a network, neither participant can deny having participated in the protocol. A non-repudiation protocol, upon successful termination, provides each participant evidence of commitment to a contract that cannot be repudiated by the other participant. A *non-repudiation of origin (NRO)* provides the recipient in an exchange, the ability to present to an adjudicator, evidence of the senders commitment to a contract. A *non-repudiation of receipt (NRR)* provides the sender in an exchange, the ability to present to an adjudicator, evidence of the recipient's commitment to a contract. An exchange protocol should satisfy the following informal requirements [19, 13]:

1. *Fairness.* The communication channels quality being fixed, at the end of the exchange protocol run, either all involved parties obtain their expected items or none (even a part) of the information to be exchanged with respect to the missing items is received.
2. *Abuse-freeness.* It is impossible for a single entity at any point in the protocol to be able to prove to an outside party that she has the power to terminate (abort) or successfully complete the protocol.
3. *Timeliness.* The communication channels quality being fixed, the parties always have the ability to reach, in a finite amount of time, a point in the protocol where they can stop the protocol while preserving fairness.

In addition to the above properties, a fair non-repudiation protocol is also expected to satisfy the following requirements: (a) *Viability.* Independently of the communication channels quality, there exists an execution of the protocol, where the exchange succeeds. (b) *Non-repudiability.* It is impossible for a single entity, after the execution of the protocol, to deny having participated in a part or the whole of the communication.

Existing protocols. Some of the existing protocols in this category are the Zhou-Gollmann (ZG) protocol [33], the Asokan-Shoup-Waidner (ASW) protocol [4], the Garay-Jakobsson-MacKenzie (GJM) protocol [13] and the Kremer-Markowitch (KM) protocol [20]. Non-repudiation protocols are difficult to design in general [32, 28, 19, 14, 16] and much literature covers the design and verification of these protocols. While some of the literature covers the discovery of vulnerabilities in these protocols based on the content of the exchanged messages, others have tried to find attacks based on the sequences of messages that can be exchanged, based on the rules of the protocols. However, there is no work that focuses on automatically obtaining correct solutions of these subtle and hard to design protocols.

Our contributions. We show that the classical synthesis formulations that are strictly competitive are inadequate for synthesizing these protocols and that newer *conditionally*

competitive formulations are more appropriate. To our knowledge this is the first application of game-theoretic controller synthesis to security protocols. Synthesis has many advantages over model checking. While model checking finds specific vulnerabilities for a designed protocol, the counter-examples in synthesis are strategies (or refinements) that exhibit vulnerabilities against a set of protocol realizations. Moreover, impossibility results such as failure to realize non-repudiation protocols without a TTP cannot be deduced with model checking, whereas such results can be deduced in a synthesis framework, as we show in this paper. Our contributions are as follows:

1. We present the formal objectives of the participants and the trusted third party as path formulas in Linear Temporal Logic (LTL) and prove that satisfaction of the objectives imply fairness of the protocols (for syntax, semantics and a description of LTL see [23, 18]). The timeliness property is also satisfied easily. The precise formalization of protocol requirements as LTL path formulas is a basic pre-requisite for synthesis.
2. We show that classical (strictly competitive) co-synthesis and weak (co-operative) co-synthesis fail, whereas assume-guarantee (conditionally competitive) co-synthesis [9] succeeds.
3. We show that all solutions in the set P_{AGS} of assume-guarantee solutions are *attack-free*, i.e., any solution in P_{AGS} prevents malicious participants from gaining an unfair advantage.
4. We show that the ASW certified mail protocol is not in P_{AGS}, due to known vulnerabilities that could have been automatically discovered. The GJM protocol is also not in P_{AGS} as it compromises our objective for the TTP, while providing fairness and abuse-freeness to the agents. The KM protocol is in P_{AGS} and it follows that it could have been automatically generated by formalizing the problem of protocol design as a synthesis problem.
5. The ASW, GJM and the KM protocol are not symmetric as they do not allow the recipient to abort the protocol. From our analysis of the refinements in P_{AGS} we construct a *new* and *symmetric* fair non-repudiation protocol that provides not just the originator but also the recipient in an exchange, the ability to abort the protocol. Given that the TTP satisfies certain constraints on her behavior, such that her objective is satisfied, we show that the symmetric protocol is attack-free.
6. Our results provide a game-theoretic justification of the need for a trusted third party. This gives an alternative justification of the impossibility results of [12, 21].

It was shown in [9] that the solutions of assume-guarantee synthesis can be obtained through the solution of secure equilibria [10] in graph games. Applying the results of [9], given our objectives, we show that for fair non-repudiation protocols, the solutions can be obtained in quadratic time.

Related works. The formal verification of fair exchange protocols uses model checking to verify a set of protocol objectives specified in a suitable temporal logic. The work of Shmatikov and Mitchell [28] uses the finite state tool Murφ to model the participants in a protocol together with an intruder model, to check a set of safety properties by state space exploration. They expose a number of vulnerabilities that may lead to replay attacks in both the ASW protocol and the GJM protocol. Zhou et al., show the use of belief logics to verify non-repudiation protocols [34]. The works [15, 14, 16, 7] use game theoretic models and the logic ATL to formally specify fairness, abuse-freeness and timeliness, that they verify using the tool MOCHA [2]. Independently, in [6] the authors use a game-based approach, with a set-rewriting technique, to verify fair exchange protocols. However, these works focus on verification and not on the synthesis of protocols. Armando et al., [3] use set-rewriting with

LTL to verify the ASW protocol and report a new attack on the protocol. Louridas in [17] provides several insightful guidelines for the design of non-repudiation protocols.

The notion of weak or *co-operative* co-synthesis was introduced in [11], classical or *strictly competitive* co-synthesis was studied in [24, 26] and assume-guarantee or *conditionally competitive* co-synthesis was introduced in [9]. But none of these works consider security protocols. The first effort at synthesizing security protocols is [22, 29] and is related to the automatic generation of mutual authentication protocols, where the authors use iterative deepening with a cost function to generate correct protocols that minimize cost; they do not address digital contract signing. In [27], the authors describe a prototype synthesis tool that uses the BAN [5] logic to describe protocol goals with extensions to describe protocol rules that, when combined with a proof system, can be used to generate protocols satisfying those goals. The authors use their approach to synthesize the Needham-Schroeder protocol; they do not address digital contract signing. The work of [1] uses multi-player games to obtain correct solutions of multi-party rational exchange protocols in the emerging area of rational cryptography. These protocols do not provide fairness, but do ensure that rational parties would have no reason to deviate from the protocol. None of the above works use a conditionally competitive synthesis formulation, which we show is necessary for fair non-repudiation protocols. Our technique is very different from these and all previous works, as we use the rich body of research in controller synthesis to construct fair exchange protocols efficiently; in time that is quadratic in the size of the model. The finite state models are typically small, so that the application of synthesis techniques as we propose in this paper is both appealing and realizable in practice.

2 Fair Non-repudiation Protocols

In this section we introduce fair non-repudiation protocols. We first define a participant model, a protocol model and an attack model. We then introduce the agents and the trusted third party that participate in fair exchange protocols, the messages that they may send and receive, and the channels over which they communicate. Finally, we introduce a set of predicates that are set based on messages that are sent and received and that form the basis for our protocol and participant objectives in the subsequent section.

A participant model. Our protocol model is different from the Strand Space model [30] and is closer to the model required for the synthesis of protocols as participant refinements. We define our model as follows: Let V be a finite set of variables that take values in some domain D_v. A *valuation* f over the variables V is a function $f : V \mapsto D_v$ that assigns to each variable $v \in V$, a value $f(v) \in D_v$; we take $\mathcal{F}[V]$ as the set of all valuations over the variables in V. Let $\mathcal{M}$ be a finite set of messages (*terms* in the Strand Space model) that are exchanged between a set $A = \{A_i \mid 0 \leq i \leq n\}$ (*roles* in the Strand Space model) of participants. We define each participant as the tuple $A_i = (L_i, V_i, \Lambda_i, \delta_i)$, where L_i is a finite set of control points or values taken by a program counter, $V_i \subseteq V$ is a set of variables, $\Lambda_i : \mathcal{F}[V_i] \mapsto 2^{\mathcal{M}}$ is a message assignment, that given a valuation $f \in \mathcal{F}[V_i]$, returns the set of messages that can be sent by A_i at f; this set includes all messages that can be composed by A_i based on what she knows in the valuation f. Valuations over variables represent what a participant knows at a given control point. We take $V = \bigcup_{i=0}^{n} V_i$ and assume that the sets V_i form a partition of V. An A_i transition function is $\delta_i : L_i \times \mathcal{F}[V_i] \times \mathcal{M} \mapsto L_i \times \mathcal{F}[V_i]$, that given a control point, a valuation over V_i and a message either sent or received by A_i, returns the next control point of A_i and an updated valuation. The participants may send messages simultaneously and independently, and can either receive a message or send a message at every control point.

The most general participants. We interpret the elements of A as the *most general participants* in an exchange; the participants in A can send any message that can be composed at each control point, based on messages they have received up to that control point. We take the interaction between the elements of A as the *most general exchange program*. Every participant in an exchange has her own objective to satisfy. We take the objective of a participant as a set of desired sequences of valuations of the protocol variables.

A protocol model. A realization of an exchange protocol is a restriction of the most general exchange program that consists of the set $A' = \{A'_i \mid 0 \leq i \leq n\}$ of participants, with behaviors restricted by the rules of the protocol. We take $A'_i = (L'_i, V_i, \Lambda'_i, \delta'_i)$, where $L'_i \subseteq L_i$; V_i is the same set of variables as in A_i; for every valuation $f \in \mathcal{F}[V_i]$ we have $\Lambda'_i(f) \subseteq \Lambda_i(f)$; and $\delta'_i : L'_i \times \mathcal{F}[V_i] \times \mathcal{M} \mapsto L'_i \times \mathcal{F}[V_i]$ is the transition function, that given a control point in L'_i, a valuation over V_i and a message either sent or received by A'_i returns the next control point of A'_i and an updated valuation. For $l \in L'_i$, $v \in \mathcal{F}[V_i]$ and $m \in \mathcal{M}$, we have $\delta'_i(l, v, m) = \delta_i(l, v, m)$. We define a *protocol instance* (or a *protocol run*) as any sequence of valuations generated by the participants in A' and take the set of all possible protocol runs as $\mathit{Runs}(A')$. We refer to a message that can be sent by a participant as a *move* of that participant.

An attack model. We define an *attack* on a protocol as the behavior of a subset of protocol participants such that the resulting sequence of messages is in their objective but not in the objective of at least one of the other participants. Formally, let $Y \subseteq A$ be a subset of the most general participants with $(A \setminus Y)' = \{A'_i | A_i \in (A \setminus Y)\}$ being the remaining participants that follow the rules of the protocol. A protocol has a Y-attack if the most general participants in Y can generate a message sequence, given $(A \setminus Y)'$ follow the protocol, that is not in the objective of at least one participant in $(A \setminus Y)'$ but is in the objectives of all participants in Y. A protocol is *attack-free*, if there exists no Y-attack for all $Y \in 2^A$.

Agents. An *agent* in a two-party exchange protocol is one of the two participating entities signing an online contract. Based on whether an agent proposes a contract or accepts a contract originating from another agent, we get two roles that an agent can play; that of an *originator* of a contract, designated by O or the *recipient* of a contract, designated by R. Agents communicate with each other over channels.

Trusted third party (TTP). The *trusted third party* or TTP is a participant who is trusted by the agents and adjudicates and resolves disputes. It is known that a fair exchange protocol cannot be realized without the TTP [12, 21]. We model the TTP explicitly as a participant, define her objective and using our formulation give a game-theoretic justification that the TTP is necessary. Agents and the TTP communicate with each other over channels.

Messages. A *message* is an encrypted stream of bytes; we treat each message as an atomic unit. We assume each message contains a *nonce* that uniquely identifies a protocol instance; participants can simultaneously participate in multiple protocol instances. We are not concerned with the exact contents of each message, but in what each message conveys; this is in keeping with our objective of synthesizing protocols that are attack-free with respect to message interleavings. From the definition of messages in fair exchange protocols in [15, 14, 16, 28] and other works, we define the set $\mathcal{M}$ of messages as follows:

- m_1 is a message that may be sent by O to R. The intent of this message is to convey O's desire to sign a contract with a recipient R.
- m_2 is a message that may be sent by R upon receiving m_1 to O. This conveys R's intent to sign the contract sent by O.

- m_3 is a message that may be sent by O to R upon receiving m_2 and contains the actual signature of O.
- m_4 is a message that contains the actual signature of R and may be sent by R to O upon receiving m_3.
- a_1^{O} is a message that may be sent by O to the TTP and conveys O's desire to *abort* the protocol.
- a_2^{O} (resp. a_2^{R}) is a message that may be sent by the TTP to O (resp. R) that confirms the abort by including an abort token for O (resp. R).
- r_1^{O} (resp. r_1^{R}) is a message that may be sent by O (resp. R) to the TTP and conveys O's (resp. R's) desire to get the TTP to *resolve* a protocol instance by explicitly requesting the TTP to adjudicate. We do not specify the content of r_1^{O} or r_1^{R} but make the assumption that the TTP needs m_1 to recover the protocol for R and similarly needs m_2 to recover the protocol for O.
- r_2^{O} (resp. r_2^{R}) is a message that may be sent by the TTP to O (resp. R) and contains a universally verifiable signature in lieu of the signature of R (resp. O).

The messages that each participant can send in a state depends on what the participant *knows* in that state. We assume that every recipient can check if the message she receives contains what she expects and that it originates from the purported sender. We impose an order on the messages m_1, m_2, m_3 and m_4 as it can be shown trivially in our synthesis formulation that O sending m_3 before receiving m_2 and R sending m_4 before receiving m_3 violates their respective objectives. Further, since our concern in this paper is not to synthesize messages impervious to attacks, but rather the correct sequences of messages that are impervious to attacks, we assume the former can be accomplished by the use of appropriate cryptographic primitives. We remark that primitives such as *private contract signatures (PCS)* introduced by Garay et al., in [13], can be used with protocols that are synthesized using our technique to ensure such properties as the *designated verifier property* which guarantees abuse-freeness. In our formulations, we consider a *reasonable TTP* that satisfies the following restrictions on behavior:

1. The TTP will never send a message unless it receives an abort or a resolve request.
2. The TTP processes messages in a first-in-first-out fashion.
3. If the first message received by the TTP is an abort request, then the TTP will eventually send an abort token.
4. If the first message received by the TTP is a resolve request, then the TTP will eventually send an agent signature.

Channels. A channel is used to deliver a *message*. There are three types of channels that are typically modeled in the literature. We present them here in decreasing order of reliability:

1. An *operational* channel delivers all messages within a known, finite amount of time.
2. A *resilient* channel eventually delivers all messages, but there is no fixed finite bound on the time to deliver a message.
3. An *unreliable* channel may not deliver all messages eventually.

We model the channels between the agents as unreliable and those between the agents and the TTP as resilient as in prevailing models; messages sent to the TTP and by the TTP will eventually be delivered. We do not model the channels explicitly, but synthesize protocols irrespective of channel behavior. In particular, unreliable channels may never deliver messages and messages sent to the TTP may arrive in any order at the TTP.

Scheduler. The scheduler is not explicitly part of any fair exchange protocol. The protocol needs to provide all agents the ability to send messages asynchronously. This implies that the

agents can choose their actions simultaneously and independently. We model this behavior by using a fair scheduler that assigns each participant a turn and we synthesize refinements against all possible behaviors of a fair scheduler.

Predicates. We introduce the following set of predicates.

- M_1 is set by O, when she sends message m_1 to R.
- EOO, referred to as the *Evidence Of Origin*, is set by R when either m_1 or r_2^{R} is received.
- EOR, referred to as the *Evidence of Receipt*, is set by O when either m_2 or r_2^{O} is received.
- $\mathrm{EOO}_k^{\mathrm{O}}$ and $\mathrm{EOO}_k^{\mathrm{TTP}}$ are referred to as *O's signature*. $\mathrm{EOO}_k^{\mathrm{O}}$ is set by R when R receives m_3 and $\mathrm{EOO}_k^{\mathrm{TTP}}$ is set by R when he receives r_2^{R}.
- $\mathrm{EOR}_k^{\mathrm{R}}$ and $\mathrm{EOR}_k^{\mathrm{TTP}}$ are referred to as *R's signature*. $\mathrm{EOR}_k^{\mathrm{R}}$ is set by O when O receives m_4 and $\mathrm{EOR}_k^{\mathrm{TTP}}$ is set by O when she receives r_2^{O}.
- AO is set by O and indicates that a_2^{O} has been received.
- AR is set by R and indicates that a_2^{R} has been received.
- ABR is set by the TTP when an abort request, a_1^{O} is received.
- RES is set by the TTP when a resolve request, r_1^{O} or r_1^{R}, is received.

All predicates are *monotonic* in that once they are set, they remain set for the duration of a protocol instance [28]. We distinguish between a signature sent by an agent and the signature sent by the TTP as a replacement for an agent's signature in the predicates. Distinguishing these signatures enables modeling TTP accountability [28]. The non-repudiation of origin for R, denoted by NRO, means that R has received both O's intent to sign a contract and O's signature on the contract so that O cannot deny having signed the contract to a third party. Formally, NRO is defined as: $\mathrm{NRO} = \mathrm{EOO} \wedge (\mathrm{EOO}_k^{\mathrm{O}} \vee \mathrm{EOO}_k^{\mathrm{TTP}})$. The non-repudiation of receipt for O, denoted by NRR, means that O has received both the intent and signature of R on a contract so that R cannot deny having signed the contract to a third party. Formally, NRR is defined as: $\mathrm{NRR} = \mathrm{EOR} \wedge (\mathrm{EOR}_k^{\mathrm{R}} \vee \mathrm{EOR}_k^{\mathrm{TTP}})$.

3 LTL Specifications for Protocol Requirements

The synthesis of programs requires a formal objective of their requirements. One of our contributions in this paper is to present a precise and formal description of the protocol requirement as a path formula in Linear Temporal Logic (LTL [23, 18]), which then becomes our synthesis objective. In this section, we define the objective for fair non-repudiation protocols, objectives for the agents and the TTP and show that satisfaction of the objectives of the agents and the TTP imply satisfaction of the objective of the protocols. We use LTL, a logic that is used to specify properties of infinite paths in finite-state transition systems. In our specifications, we use the usual LTL notations $\Box$ and $\Diamond$ to denote *always* (safety) and *eventually* (reachability) specifications, respectively.

Fairness. Informally, fairness for O can be stated as *"For all protocol instances if the non-repudiation of origin (NRO) is ever true, then eventually the non-repudiation of receipt (NRR) is also true"* [16]. The fairness property for O is expressed by the LTL formula

$$\varphi_f^{\mathrm{O}} = \Box(\mathrm{NRO} \Rightarrow \Diamond\mathrm{NRR}) \ .$$

Similarly, the fairness property for R is expressed by the LTL formula $\varphi_f^{\mathrm{R}} = \Box(\mathrm{NRR} \Rightarrow \Diamond\mathrm{NRO})$. We say that a protocol is fair, if in all instances of the protocol, fairness for both O and R holds. Hence the fairness requirement for the protocol is expressed by the formula

$$\varphi_f = \varphi_f^{\mathrm{O}} \wedge \varphi_f^{\mathrm{R}} \ . \tag{1}$$

Abuse-freeness. The definition of abuse-freeness as given in [13], is the following: *"An optimistic contract signing protocol is abuse-free if it is impossible for a single player at any point in the protocol to be able to prove to an outside party that he has the power to terminate (abort) or successfully complete the contract"*. In [8], the authors prove that in any fair optimistic protocol, an optimistic participant yields an advantage to the other participant. In a given protocol instance, once an agent has the other agent's intent to sign a contract, he can use this intent to negotiate a different contract with a third party, while ensuring that the original protocol instance is aborted. The term aborted is used here to mean that neither agent can get a non-repudiation evidence in a given protocol instance, once that instance is aborted. As noted by the authors of [8], the best that one can hope for is to prevent either participant from proving to a third party that he has an advantage, or in other words, that he has the other participant's intent to sign the contract. This is defined as *abuse-freeness*. As noted by the authors of [13, 15], using PCS or *Private Contract Signatures*, introduced by Garay et al., in [13], which provides the designated verifier property, neither agent can prove the other agent's intent to sign the contract to anyone other than the TTP. Therefore, ensuring abuse-freeness requires the use of PCS. Since PCS are requisite to ensure abuse-freeness, we do not model abuse-freeness, or the stronger property balance [6], in our formalism.

Timeliness. Informally, timeliness is defined as follows: *"A protocol respects timeliness, if both agents always have the ability to reach, in a finite amount of time, a point in the protocol where they can stop the protocol while preserving fairness"*. We do not model timeliness in this paper as the cases in the literature where timeliness is compromised involve the lack of an abort subprotocol. Since we explicitly include the capability to abort the protocol, our solution provides timeliness as guaranteed by existing protocols. Alternatively, timeliness could be explicitly modeled in the specifications of the agents and the TTP, but in the interest of keeping the objectives simpler so that we convey the more interesting idea of using assume-guarantee synthesis, we avoid modeling timeliness explicitly.

Signature exchange. A protocol is an exchange protocol if it enables the exchange of signatures. This is also referred to as *Viability* in the literature. For an exchange protocol to be a non-repudiation protocol, at the end of every run of the protocol, either the agents have their respective non-repudiation evidences, or, if they do not have their non-repudiation evidences, they have the abort token. The property that evidences once obtained are not repudiable is referred to as *Non-repudiability*. A fair non-repudiation protocol must satisfy fairness, abuse-freeness, non-repudiability and viability.

We now present intuitive objectives for the agents and the trusted third party and show that satisfaction of these objectives implies that the protocols we synthesize are fair.

Specification for the originator O. The objective of the originator O is expressed as follows:

- In all protocol instances, she eventually sends the evidence of origin. This is expressed by the LTL formula $\varphi^1_{\mathrm{O}} = \Diamond M_1$.
- In all protocol instances, one of the following statements should be true:
  1. (a) The originator eventually gets the recipient's signature $\mathrm{EOR}^{\mathrm{R}}_k$ or, (b) she eventually gets the recipient's signature $\mathrm{EOR}^{\mathrm{TTP}}_k$ and never gets the abort token AO. This is expressed by the LTL formula

$$\varphi^2_{\mathrm{O}} = (\Diamond \mathrm{EOR}^{\mathrm{R}}_k \vee (\Diamond \mathrm{EOR}^{\mathrm{TTP}}_k \wedge \Box \neg \mathrm{AO}))\ .$$

2. (a) The originator eventually gets the abort token and, (b) the recipient never gets her signature $\mathrm{EOO}_k^{\mathrm{O}}$ and never gets her signature $\mathrm{EOO}_k^{\mathrm{TTP}}$ from the TTP. This is expressed by the LTL formula

$$\varphi_{\mathrm{O}}^3 = \Diamond \mathrm{AO} \wedge (\Box \neg \mathrm{EOO}_k^{\mathrm{O}} \wedge \Box \neg \mathrm{EOO}_k^{\mathrm{TTP}}) = \Diamond \mathrm{AO} \wedge \Box (\neg \mathrm{EOO}_k^{\mathrm{O}} \wedge \neg \mathrm{EOO}_k^{\mathrm{TTP}}) \ .$$

The objective φ_{O} of O can therefore be expressed by the following LTL formula

$$\varphi_{\mathrm{O}} = \varphi_{\mathrm{O}}^1 \wedge \Box (\varphi_{\mathrm{O}}^2 \vee \varphi_{\mathrm{O}}^3) \ . \tag{2}$$

There are two interpretations of the abort token in the literature. On the one hand the abort token was never intended to serve as a proof that a protocol instance was not successfully completed; it was to guarantee that the TTP would never resolve a protocol after it has been aborted. On the other hand, there is mention of the abort token being used by the recipient to prove that the protocol was aborted. We take the position that the abort token may be used to ensure TTP accountability as noted in [28] and hence include it in the objective of O. If the TTP misbehaves and issues both $\mathrm{EOR}_k^{\mathrm{TTP}}$ and AO, we claim that the objective φ_{O} of the originator should be violated, but in this case, she has the power to prove that the TTP misbehaved by presenting both $\mathrm{EOR}_k^{\mathrm{TTP}}$ and AO to demonstrate inconsistent behavior. While having both $\mathrm{EOR}_k^{\mathrm{TTP}}$ and AO is a violation of φ_{O}, having both $\mathrm{EOR}_k^{\mathrm{R}}$ and AO is not a violation of φ_{O}; once O receives $\mathrm{EOR}_k^{\mathrm{R}}$, we take it that the objective φ_{O} is satisfied. While having both $\mathrm{EOR}_k^{\mathrm{R}}$ and $\mathrm{EOR}_k^{\mathrm{TTP}}$ may be interpreted as O having inconsistent signatures, we do not consider this to be a violation of O's objective; given the nature of asynchronous networks it may well be the case that both these evidences arrive eventually, one from the TTP and the other from R, as O did not wait long enough before sending r_1^{O}.

Specification for the recipient R. The objective of the recipient R can be expressed as follows:

– In all protocol instances, if he gets the evidence of origin EOO, then one of the following statements should be true:
  1. (a) The recipient eventually gets the originator's signature $\mathrm{EOO}_k^{\mathrm{O}}$ or, (b) he eventually gets the originator's signature $\mathrm{EOO}_k^{\mathrm{TTP}}$ and never gets the abort token AR. This is expressed by the LTL formula

$$\varphi_{\mathrm{R}}^1 = (\Diamond \mathrm{EOO}_k^{\mathrm{O}} \vee (\Diamond \mathrm{EOO}_k^{\mathrm{TTP}} \wedge \Box \neg \mathrm{AR})) \ .$$

  2. (a) The recipient eventually gets the abort token and, (b) the originator never gets his signature $\mathrm{EOR}_k^{\mathrm{R}}$ and never gets his signature $\mathrm{EOR}_k^{\mathrm{TTP}}$ from the TTP. This is expressed by the LTL formula

$$\varphi_{\mathrm{R}}^2 = \Diamond \mathrm{AR} \wedge (\Box \neg \mathrm{EOR}_k^{\mathrm{R}} \wedge \Box \neg \mathrm{EOR}_k^{\mathrm{TTP}}) = \Diamond \mathrm{AR} \wedge \Box (\neg \mathrm{EOR}_k^{\mathrm{R}} \wedge \neg \mathrm{EOR}_k^{\mathrm{TTP}}) \ .$$

The objective φ_{R} can therefore be expressed by the LTL formula

$$\varphi_{\mathrm{R}} = \Box (\mathrm{EOO} \Rightarrow (\varphi_{\mathrm{R}}^1 \vee \varphi_{\mathrm{R}}^2)) \ . \tag{3}$$

If the TTP misbehaves and issues both $\mathrm{EOO}_k^{\mathrm{TTP}}$ and AR, we claim that the objective φ_{R} of the recipient should be violated, but in this case he has the power to prove that the TTP misbehaved by presenting both $\mathrm{EOO}_k^{\mathrm{TTP}}$ and AR. Similar to the case of O, once R receives $\mathrm{EOO}_k^{\mathrm{O}}$, the objective φ_{R} is satisfied whether or not abort tokens or non-repudiation evidences are issued by the TTP.

Specification for the trusted third party TTP. The objective of the trusted third party is to treat both agents symmetrically and be accountable to both agents. It can be expressed as follows:

– In all protocol instances, if the abort request a_1^{O} or a resolve request r_1^{O} or r_1^{R} is received, then eventually the TTP sends the abort token AO or the abort token AR or the originator's signature $\mathrm{EOO}_k^{\mathrm{TTP}}$ or the recipient's signature $\mathrm{EOR}_k^{\mathrm{TTP}}$. This can be expressed by the LTL formula

$$\varphi_{\mathrm{TTP}}^1 = \Box((\mathrm{ABR} \vee \mathrm{RES}) \Rightarrow (\Diamond \mathrm{AO} \vee \Diamond \mathrm{AR} \vee \Diamond \mathrm{EOO}_k^{\mathrm{TTP}} \vee \Diamond \mathrm{EOR}_k^{\mathrm{TTP}})) .$$

– In all protocol instances, if the originator's signature $\mathrm{EOO}_k^{\mathrm{TTP}}$ has been sent to the recipient, then the originator should eventually get the recipient's signature $\mathrm{EOR}_k^{\mathrm{TTP}}$ and the agents should never get the abort token. This can be expressed by the LTL formula

$$\varphi_{\mathrm{TTP}}^2 = \Box(\mathrm{EOO}_k^{\mathrm{TTP}} \Rightarrow (\Diamond \mathrm{EOR}_k^{\mathrm{TTP}} \wedge \Box(\neg \mathrm{AO} \wedge \neg \mathrm{AR}))) .$$

– Symmetrically, in all protocol instances, if the recipient's signature $\mathrm{EOR}_k^{\mathrm{TTP}}$ has been sent to the originator, then the recipient should eventually get the originator's signature $\mathrm{EOO}_k^{\mathrm{TTP}}$ and the agents should never get the abort token. This can be expressed by the LTL formula

$$\varphi_{\mathrm{TTP}}^3 = \Box(\mathrm{EOR}_k^{\mathrm{TTP}} \Rightarrow (\Diamond \mathrm{EOO}_k^{\mathrm{TTP}} \wedge \Box(\neg \mathrm{AO} \wedge \neg \mathrm{AR}))) .$$

– In all protocol instances, if the originator gets the abort token AO, then the recipient should eventually get the abort token AR and the originator should never get the recipient's signature $\mathrm{EOR}_k^{\mathrm{TTP}}$ and the recipient should never get the originator's signature $\mathrm{EOO}_k^{\mathrm{TTP}}$. This can be expressed by the LTL formula

$$\varphi_{\mathrm{TTP}}^4 = \Box(\mathrm{AO} \Rightarrow (\Diamond \mathrm{AR} \wedge \Box(\neg \mathrm{EOO}_k^{\mathrm{TTP}} \wedge \neg \mathrm{EOR}_k^{\mathrm{TTP}}))) .$$

– Symmetrically, in all protocol instances, if the recipient gets the abort token AR, then the originator should eventually get the abort token AO and the originator should never get the recipient's signature $\mathrm{EOR}_k^{\mathrm{TTP}}$ and the recipient should never get the originator's signature $\mathrm{EOO}_k^{\mathrm{TTP}}$. This can be expressed by the LTL formula

$$\varphi_{\mathrm{TTP}}^5 = \Box(\mathrm{AR} \Rightarrow (\Diamond \mathrm{AO} \wedge \Box(\neg \mathrm{EOO}_k^{\mathrm{TTP}} \wedge \neg \mathrm{EOR}_k^{\mathrm{TTP}}))) .$$

The objective φ_{TTP} of the TTP is then defined as:

$$\varphi_{\mathrm{TTP}} = \varphi_{\mathrm{TTP}}^1 \wedge \varphi_{\mathrm{TTP}}^2 \wedge \varphi_{\mathrm{TTP}}^3 \wedge \varphi_{\mathrm{TTP}}^4 \wedge \varphi_{\mathrm{TTP}}^5 . \tag{4}$$

Note that our objective for the TTP treats both agents symmetrically. In this paper we present assume-guarantee synthesis for the above objective of the TTP. But in general, the objective of the TTP can be weakened if desired, by treating the agents asymmetrically, and the assume-guarantee synthesis technique can be applied with this weakened objective. We remark that the specifications of the participants in our protocol model are sequences of messages. Using predicates that are set when messages are sent or received by the agents or the TTP, we transform those informal specifications into formal objectives using the predicates and LTL. The following theorem shows that the objectives we have introduced (2), (3) and (4) imply fairness (1).

Theorem 1 (Objectives imply fairness) *We have,* $\varphi_O \wedge \varphi_R \wedge \varphi_{TTP} \Rightarrow \varphi_f$.

Proof To prove the assertion, assume towards a contradiction that there exists a path that satisfies $\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}$ but does not satisfy φ_f. We consider the case when the path does not satisfy the first conjunct $\varphi_f^{\mathrm{O}} = \Box(\mathrm{NRO} \Rightarrow \Diamond\mathrm{NRR})$ (a similar argument applies to the second conjunct). If the path does not satisfy φ_f^{O}, then there is a suffix of the path, where $\mathrm{EOO} \wedge (\mathrm{EOO}_k^{\mathrm{O}} \vee \mathrm{EOO}_k^{\mathrm{TTP}})$ holds but $\mathrm{EOR} \wedge (\mathrm{EOR}_k^{\mathrm{R}} \vee \mathrm{EOR}_k^{\mathrm{TTP}})$ does not hold at all states of the suffix. It follows that the path satisfies

$$\Diamond\Box(\mathrm{EOO} \wedge (\mathrm{EOO}_k^{\mathrm{O}} \vee \mathrm{EOO}_k^{\mathrm{TTP}}) \wedge (\neg\mathrm{EOR} \vee (\neg\mathrm{EOR}_k^{\mathrm{R}} \wedge \neg\mathrm{EOR}_k^{\mathrm{TTP}}))) \ . \tag{5}$$

Consider the objective $\varphi_{\mathrm{O}}^2 = (\Diamond\mathrm{EOR}_k^{\mathrm{R}} \vee (\Diamond\mathrm{EOR}_k^{\mathrm{TTP}} \wedge \Box\neg\mathrm{AO}))$. Since all predicates are monotonic, we can rewrite φ_{O}^2 as follows:

$$\varphi_{\mathrm{O}}^2 = \Diamond\Box(\mathrm{EOR}_k^{\mathrm{R}} \vee (\mathrm{EOR}_k^{\mathrm{TTP}} \wedge \neg\mathrm{AO})) \ .$$

Similarly, we can rewrite φ_{O}^3 as follows:

$$\varphi_{\mathrm{O}}^3 = \Diamond\Box(\mathrm{AO} \wedge \neg\mathrm{EOO}_k^{\mathrm{O}} \wedge \neg\mathrm{EOO}_k^{\mathrm{TTP}}) \ .$$

If a path satisfies (5), then it also satisfies $\Diamond\Box(\mathrm{EOO}_k^{\mathrm{O}} \vee \mathrm{EOO}_k^{\mathrm{TTP}})$. By the monotonicity of the predicates, we have $\Diamond\Box(\mathrm{EOO}_k^{\mathrm{O}} \vee \mathrm{EOO}_k^{\mathrm{TTP}})$ is equivalent to $\Diamond\Box\mathrm{EOO}_k^{\mathrm{O}} \vee \Diamond\Box\mathrm{EOO}_k^{\mathrm{TTP}}$. We consider the following cases to complete the proof:

1. *Case 1. Path satisfies $\Diamond\Box EOO_k^{O}$.* If the path satisfies $\Diamond\Box\mathrm{EOO}_k^{\mathrm{O}}$, then the path does not satisfy φ_{O}^3. We now show that the path also does not satisfy φ_{O}^2. Since the path satisfies $\Diamond\Box\mathrm{EOO}_k^{\mathrm{O}}$, it must be the case that message m_2 was received by O, as otherwise O will not send $\mathrm{EOO}_k^{\mathrm{O}}$. This implies that the path satisfies $\Diamond\Box\mathrm{EOR}$. Since the path satisfies both $\Diamond\Box\mathrm{EOR}$ and (5), it follows that the path must satisfy $\Diamond\Box(\neg\mathrm{EOR}_k^{\mathrm{R}} \wedge \neg\mathrm{EOR}_k^{\mathrm{TTP}})$. Hence the path does not satisfy $\Diamond\Box\mathrm{EOR}_k^{\mathrm{R}}$ and $\Diamond\Box\mathrm{EOR}_k^{\mathrm{TTP}}$ leading to the path violating φ_{O}^2. Since the path does not satisfy both φ_{O}^2 and φ_{O}^3, it does not satisfy φ_{O}, which is a contradiction.
2. *Case 2. Path satisfies $\Diamond\Box EOO_k^{TTP}$.* If the path satisfies $\Diamond\Box\mathrm{EOO}_k^{\mathrm{TTP}}$, then either O or R must have sent the resolve request. If the TTP resolves the protocol only to the agent that sends the resolve request and not the other, then the path does not satisfy φ_{TTP}, leading to a contradiction. For φ_{TTP} to hold, the TTP must have sent both $\mathrm{EOO}_k^{\mathrm{TTP}}$ and $\mathrm{EOR}_k^{\mathrm{TTP}}$, which given the channels between the agents and the TTP are resilient implies, (a) EOR must have been set by O upon receiving $\mathrm{EOR}_k^{\mathrm{TTP}}$ leading to the path satisfying $\Diamond\Box\mathrm{EOR}$ and (b) the path satisfies $\Diamond\Box\mathrm{EOR}_k^{\mathrm{TTP}}$. Since the path satisfies $\Diamond\Box\mathrm{EOR}$ and $\Diamond\Box\mathrm{EOR}_k^{\mathrm{TTP}}$, it cannot satisfy (5), leading to a contradiction.

■

4 Co-synthesis

In this section we first define processes, schedulers and objectives for synthesis along the lines of [9]. Next we define traditional co-operative [11] and strictly competitive [24, 25] versions of the co-synthesis problem; we refer to them as *weak co-synthesis* and *classical co-synthesis*, respectively. We then define a formulation of co-synthesis introduced in [9] called *assume-guarantee synthesis*. We show later in the paper that the protocol model of Section 2 reduces to the process model for synthesis that we present in this section.

Variables, valuations, and traces. Let X be a finite set of variables such that each variable $x \in X$ has a finite domain D_x. A *valuation* f on X is a function $f : X \to \bigcup_{x\in X} D_x$

that assigns to each variable $x \in X$ a value $f(x) \in D_x$. We write $\mathcal{F}[X]$ for the set of valuations on X. A *trace* on X is an infinite sequence $(v_0, v_1, v_2, \ldots) \in \mathcal{F}[X]^\omega$ of valuations on X. Given a valuation $f[X] \in \mathcal{F}[X]$ and a subset $Y \subseteq X$ of the variables, we denote by $f[X] \downarrow Y$ the restriction of the valuation $f[X]$ to the variables in Y. Similarly, for a trace $\tau(X) = (v_0, v_1, v_2, \ldots)$ on X, we write $\tau(X) \downarrow Y = (v_0 \downarrow Y, v_1 \downarrow Y, v_2 \downarrow Y, \ldots)$ for the restriction of $\tau(X)$ to the variables in Y. The restriction operator is lifted to sets of valuations, and to sets of traces.

Processes and refinement. Let *Moves* be a finite set of moves. For $i \in \{1, 2, 3\}$, a *process* is defined by the tuple $P_i = (X_i, \Gamma_i, \delta_i)$ where,

1. X_i is a finite set of variables of process P_i with $X = \bigcup_{i=1}^{3} X_i$ being the set of all process variables,
2. $\Gamma_i : \mathcal{F}_i[X_i] \to 2^{Moves} \setminus \emptyset$ is a move assignment that given a valuation in $\mathcal{F}_i[X_i]$, returns a non-empty set of moves, where $\mathcal{F}_i[X_i]$ is the set of valuations on X_i, and
3. $\delta_i : \mathcal{F}_i[X_i] \times \mathit{Moves} \to 2^{\mathcal{F}_i[X_i]} \setminus \emptyset$ is a non-deterministic transition function.

The set of process variables X may be shared between processes. The processes only choose amongst available moves at every valuation of their variables as determined by their move assignment. The transition function maps a present valuation and a process move to a nonempty set of possible successor valuations such that each successor valuation has a unique pre-image. The uniqueness of the pre-image is a property of fair exchange protocols; unique messages convey unique content and generate unique valuations.

A *refinement* of process $P_i = (X_i, \Gamma_i, \delta_i)$ is a process $P'_i = (X'_i, \Gamma'_i, \delta'_i)$ such that:

1. $X_i \subseteq X'_i$,
2. for all valuations $f_i[X'_i]$ on X'_i, we have $\Gamma'_i(f_i[X'_i]) \subseteq \Gamma_i(f_i[X'_i] \downarrow X_i)$, and
3. for all valuations $f_i[X'_i]$ on X'_i and for all moves $a \in \Gamma'_i(f_i[X'_i])$, we have $\delta'_i(f_i[X'_i], a) \downarrow X_i \subseteq \delta_i(f_i[X'_i] \downarrow X_i, a)$.

In other words, the refined process P'_i has possibly more variables than the original process P_i, at most the same moves as the moves of the original process P_i at every valuation, and every possible update of the variables in X_i given Γ'_i by P'_i is a possible update by P_i. We write $P'_i \preceq P_i$ to denote that P'_i is a refinement of P_i. Given refinements P'_1 of P_1, P'_2 of P_2 and P'_3 of P_3, we write $X' = X'_1 \cup X'_2 \cup X'_3$ for the set of variables of all refinements, and we denote the set of valuations on X' by $\mathcal{F}[X']$.

Schedulers. Given processes P_i, where $i \in \{1, 2, 3\}$, a *scheduler* Sc for P_i chooses at each computation step whether it is process P_1's turn, process P_2's turn or process P_3's turn to update her variables. Formally, the scheduler Sc is a function $\mathrm{Sc} : \mathcal{F}[X]^* \to \{1, 2, 3\}$ that maps every finite sequence of global valuations (representing the history of a computation) to $i \in \{1, 2, 3\}$, signaling that process P_i is next to update her variables. The scheduler Sc is *fair* if it assigns turns to P_1, P_2 and P_3 infinitely often; i.e., for all traces $(v_0, v_1, v_2, \ldots) \in \mathcal{F}[X]^\omega$, there exist infinitely many $j_i \geq 0$, such that $\mathrm{Sc}(v_0, \ldots, v_{j_1}) = 1$, $\mathrm{Sc}(v_0, \ldots, v_{j_2}) = 2$ and $\mathrm{Sc}(v_0, \ldots, v_{j_3}) = 3$. Given three processes $P_1 = (X_1, \Gamma_1, \delta_1)$, $P_2 = (X_2, \Gamma_2, \delta_2)$ and $P_3 = (X_3, \Gamma_3, \delta_3)$, a scheduler Sc for P_1, P_2 and P_3, and a start valuation $v_0 \in \mathcal{F}[X]$, the set of possible traces is:

$$\begin{aligned}
[\![(P_1 \parallel P_2 \parallel P_3 \parallel \mathrm{Sc})(v_0)]\!] = \{(v_0, v_1, v_2, \ldots) \in \mathcal{F}[X]^\omega \mid {} & \forall j \geq 0.\ \mathrm{Sc}(v_0, \ldots, v_j) = i; \\
& v_{j+1} \downarrow (X \setminus X_i) = v_j \downarrow (X \setminus X_i); \\
& v_{j+1} \downarrow X_i \in \delta_i(v_j \downarrow X_i, a) \text{ for some } a \in \Gamma_i(v_j \downarrow X_i))\} .
\end{aligned}$$

Note that during turns of one process P_i, the values of the private variables $X \setminus X_i$ of the other processes remain unchanged. We define the projection of traces to moves as follows:

$$(v_0, v_1, v_2, \ldots) \downarrow \mathit{Moves} = \{(a_0, a_1, a_2, \ldots) \in \mathit{Moves}^\omega \mid \forall j \geq 0.\ \mathrm{Sc}(v_0, \ldots, v_j) = i;\\ v_{j+1} \downarrow X_i \in \delta_i(v_j \downarrow X_i, a_j); a_j \in \Gamma_i(v_j \downarrow X_i)\}\ .$$

Specifications. A *specification* φ_i for process P_i is a set of traces on X; that is, $\varphi_i \subseteq \mathcal{F}[X]^\omega$. We consider only ω-regular specifications [31]. We define boolean operations on specifications using logical operators such as $\wedge$ (conjunction) and $\Rightarrow$ (implication).

The input to the co-synthesis problem is given as follows: for $i \in \{1, 2, 3\}$, processes $P_i = (X_i, \Gamma_i, \delta_i)$, specifications φ_i for process i, and a start valuation $v_0 \in \mathcal{F}$.

Weak co-synthesis. The *weak co-synthesis* problem is defined as follows: do there exist refinements $P_i' = (X_i', \Gamma_i', \delta_i')$ and a valuation $v_0' \in \mathcal{F}'$, such that,

1. $P_i' \preceq P_i$ and $v_0' \downarrow X = v_0$, and
2. For all fair schedulers Sc for P_i' we have, $[\![(P_1' \parallel P_2' \parallel P_3' \parallel \mathrm{Sc})(v_0')]\!] \downarrow X \subseteq (\varphi_1 \wedge \varphi_2 \wedge \varphi_3)$.

Intuitively, weak co-synthesis or co-operative co-synthesis is a synthesis formulation that seeks refinements P_1', P_2' and P_3' where the processes co-operate to satisfy their respective objectives.

Classical co-synthesis. The *classical co-synthesis* problem is defined as follows: do there exist refinements $P_i' = (X_i', \Gamma_i', \delta_i')$ and a valuation $v_0' \in \mathcal{F}'$, such that,

1. $P_i' \preceq P_i$ and $v_0' \downarrow X = v_0$, and
2. For all fair schedulers Sc for P_i' we have,
   (a) $[\![(P_1' \parallel P_2 \parallel P_3 \parallel \mathrm{Sc})(v_0')]\!] \downarrow X \subseteq \varphi_1$;
   (b) $[\![(P_1 \parallel P_2' \parallel P_3 \parallel \mathrm{Sc})(v_0')]\!] \downarrow X \subseteq \varphi_2$;
   (c) $[\![(P_1 \parallel P_2 \parallel P_3' \parallel \mathrm{Sc})(v_0')]\!] \downarrow X \subseteq \varphi_3$.

Classical or strictly competitive co-synthesis is a formulation that seeks refinements P_1', P_2' and P_3' such that P_1' can satisfy φ_1 against all possible, and hence adversarial, behaviors of the other processes; similarly for P_2' and P_3'.

Assume-guarantee synthesis. The *assume-guarantee synthesis* problem is defined as follows: do there exist refinements $P_i' = (X_i', \Gamma_i', \delta_i')$ and a valuation $v_0' \in \mathcal{F}'$, such that,

1. $P_i' \preceq P_i$ and $v_0' \downarrow X = v_0$, and
2. For all fair schedulers Sc for P_i' we have,
   (a) $[\![(P_1' \parallel P_2 \parallel P_3 \parallel \mathrm{Sc})(v_0')]\!] \downarrow X \subseteq (\varphi_2 \wedge \varphi_3) \Rightarrow \varphi_1$;
   (b) $[\![(P_1 \parallel P_2' \parallel P_3 \parallel \mathrm{Sc})(v_0')]\!] \downarrow X \subseteq (\varphi_1 \wedge \varphi_3) \Rightarrow \varphi_2$;
   (c) $[\![(P_1 \parallel P_2 \parallel P_3' \parallel \mathrm{Sc})(v_0')]\!] \downarrow X \subseteq (\varphi_1 \wedge \varphi_2) \Rightarrow \varphi_3$;
   (d) $[\![(P_1' \parallel P_2' \parallel P_3' \parallel \mathrm{Sc})(v_0')]\!] \downarrow X \subseteq (\varphi_1 \wedge \varphi_2 \wedge \varphi_3)$.

Assume-guarantee synthesis or conditionally competitive co-synthesis is a formulation that seeks refinements P_1', P_2' and P_3' such that P_1' can satisfy φ_1 as long as processes P_2 and P_3 satisfy their objectives; similarly for P_2' and P_3'. This synthesis formulation is well suited for those cases where processes are primarily concerned with satisfying their own objectives and only secondarily concerned with violating the objectives of the other processes. We want protocols to be correct under *arbitrary* behaviors of the participants, and the arbitrary or worst case behavior of a participant without sabotaging her own objective, is to first

satisfy her own objective, and only then to falsify the objectives of the other participants. The primary goal of satisfying her own objective, and secondary goal of falsifying other participant objectives formally captures this worst case or arbitrary behavior assumption. We show that this synthesis formulation is the only one that works for fair non-repudiation protocols. While classical co-synthesis can be solved as zero-sum games, assume-guarantee synthesis can be solved using non zero-sum games with lexicographic objectives [9]. For brevity, we drop the initial valuation v_0 in the set of traces.

5 Protocol Co-synthesis

In this section, we present our results on synthesizing fair non-repudiation protocols. We use the process model in Section 4 to define agent and TTP processes, with objectives as defined in Section 3. We then introduce the protocol synthesis model and show that classical co-synthesis fails and weak co-synthesis generates unacceptable solutions. We provide a game theoretic justification of the need for a TTP by showing that without the TTP neither classical co-synthesis nor assume-guarantee synthesis can be used to synthesize fair non-repudiation protocols. We define the set P_{AGS} of assume-guarantee refinements and prove that the refinements are attack-free. We then present an alternate characterization of the set P_{AGS} and show that the Kremer-Markowitch (KM) non-repudiation protocol with offline TTP, proposed in [20, 14, 16], is included in P_{AGS} whereas the ASW certified mail protocol and the GJM protocol are not. Finally, we systematically analyze refinements of the most general agents and the TTP with respect to their membership in P_{AGS} and show the KM protocol can be automatically generated.

The process O. We distinguish between the set of messages sent by O and the set of messages received by O. We first recall, from Section 2, that O sets the predicates EOR, $\mathrm{EOR}_k^{\mathrm{R}}$, $\mathrm{EOR}_k^{\mathrm{TTP}}$ and AO when she receives messages m_2, m_4, r_2^{O} and a_2^{O} respectively. We add to this set the predicates M_1, M_3, $\mathrm{ABR}^{\mathrm{O}}$ and $\mathrm{RES}^{\mathrm{O}}$ that are set by O when she sends messages m_1, m_3, a_1^{O} and r_1^{O} respectively. We take the set of variables of the process O as $X_{\mathrm{O}} = \{M_1, \mathrm{EOR}, M_3, \mathrm{EOR}_k^{\mathrm{R}}, \mathrm{EOR}_k^{\mathrm{TTP}}, \mathrm{ABR}^{\mathrm{O}}, \mathrm{RES}^{\mathrm{O}}, \mathrm{AO}\}$; the union of the predicates set by O when she receives messages and the set of predicates set by O when she sends messages. By an abuse of notation, we take the set of all messages that can be sent by O as the moves of process O. By including an idle move ι, which O may choose in lieu of sending a message, we get the following set of moves for O: $\mathit{Moves}_{\mathrm{O}} = \{\iota, m_1, m_3, a_1^{\mathrm{O}}, r_1^{\mathrm{O}}\}$.

The process R. Similar to the case of process O, we define the set of variables of process R as the union of the set of predicates set by R when he sends messages and the set of predicates he sets when he receives messages. We have the predicates EOO, $\mathrm{EOO}_k^{\mathrm{O}}$, $\mathrm{EOO}_k^{\mathrm{TTP}}$ and AR, set by R when he receives messages m_1, m_3, r_2^{R} and a_2^{R} respectively. We add to this the predicates M_2, M_4 and $\mathrm{RES}^{\mathrm{R}}$, set by R when he sends messages m_2, m_4 and r_1^{R} respectively. This gives us the following variables for process R: $X_{\mathrm{R}} = \{\mathrm{EOO}, M_2, \mathrm{EOO}_k^{\mathrm{O}}, M_4, \mathrm{EOO}_k^{\mathrm{TTP}}, \mathrm{RES}^{\mathrm{R}}, \mathrm{AR}\}$. The set of moves for R is given by $\mathit{Moves}_{\mathrm{R}} = \{\iota, m_2, m_4, r_1^{\mathrm{R}}\}$. In Figure 1, we show an interface automaton for an agent. Since an agent can act either as an originator or a recipient, we show the actions available to the agent in both roles in the figure.

The process TTP. The predicates ABR and RES are set by the TTP when she receives an abort or a resolve request from either agent. We add to this the predicates A_2^{O}, A_2^{R}, R_2^{O} and R_2^{R}, set by the TTP when she sends messages a_2^{O}, a_2^{R}, r_2^{O} and r_2^{R} respectively. We get the following set of process variables for the TTP: $X_{\mathrm{TTP}} = \{\mathrm{ABR}, \mathrm{RES}, A_2^{\mathrm{O}}, A_2^{\mathrm{R}}, R_2^{\mathrm{O}}, R_2^{\mathrm{R}}\}$. The set of moves for the TTP are defined as follows: $\mathit{Moves}_{\mathrm{TTP}} = \{\iota, a_2^{\mathrm{O}}, a_2^{\mathrm{R}}, [a_2^{\mathrm{O}}, a_2^{\mathrm{R}}], r_2^{\mathrm{O}}, r_2^{\mathrm{R}}, [r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]\}$. The TTP move $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$ results in the

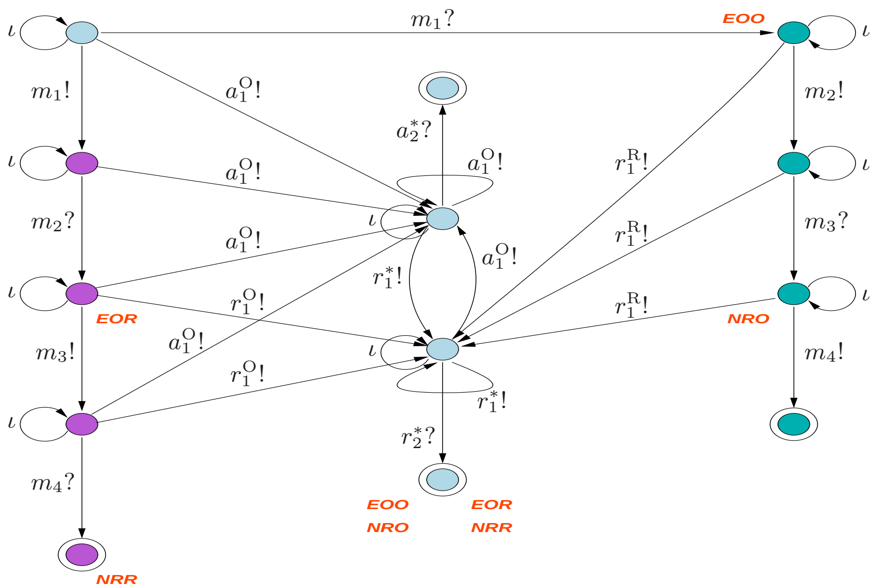

Fig. 1: An interface automaton that shows the states and enabled moves of the agents O (on the left) and R (on the right). Move ι is the idle move. The states with no outgoing edges are terminal. We consider the most liberal behaviors of the agents wherein the abort and resolve messages can be sent from all states where the agents have the data they need to send those messages. The predicates are monotonic and are shown in the first state at which they hold. In states that can be either agent state, we use the * in the messages a_2^*, r_1^*, r_2^* to denote one of O or R. Abort or resolve requests can be sent from the states marked terminal, but they have no bearing on the outcome of the protocol and hence we omit them.

TTP sending messages a_2^{O} to O and a_2^{R} to R. The TTP can choose to send them in any order; all that is guaranteed is that both messages will be sent by the TTP. Similarly for the TTP move $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$. The moves for the TTP are shown in Table 1; these include the moves for the TTP in the ASW certified mail protocol [4], the GJM protocol [13] and the KM protocol [20]. We show moves for the TTP with and without a persistent database for completeness. Since it is trivially the case that TTP accountability cannot be satisfied without a persistent database, we do not consider the absence of a persistent database in the rest of this paper.

The protocol synthesis model. We now have all the ingredients to define our protocol synthesis model. Given process O, process R and process TTP as defined above, we take $X = X_{\mathrm{O}} \cup X_{\mathrm{R}} \cup X_{\mathrm{TTP}}$ as the joint set of process variables. We take the objectives φ_{O}, φ_{R} and φ_{TTP} for the processes O, R and TTP respectively, as defined in Section 3. The set of traces $[\![\mathrm{O} \parallel \mathrm{R} \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!]$, given Sc is a fair scheduler, is then the joint behavior of the most general agents and the most general TTP, subject to the constraint that they can only send messages based on what they know at every valuation of their variables. A protocol is a refinement $\mathrm{O}' \preceq \mathrm{O}$, $\mathrm{R}' \preceq \mathrm{R}$ and $\mathrm{TTP}' \preceq \mathrm{TTP}$, where each participant has a restricted set of moves at every valuation of the process variables; the restrictions constituting the rules of the protocol. We take a protocol state as a valuation over the process variables. By an abuse of notation, we represent every state of the protocol by the set of variables that are set to true in that state; for example a valuation $f = \{M_1, \mathrm{EOO}, M_2, \mathrm{EOR}\}$ corresponds to the

Agent moves	Enabled TTP moves				
	Without DB		With a persistent DB		
			ASW	GJM	KM
O sends a_1^{O}	a_2^{O}	$[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$	If R has recovered, invite O to recover else a_2^{O}	If recovered, then r_2^{O} else a_2^{O}	If aborted or recovered, then ι else $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$
O sends r_1^{O}	r_2^{O}	$[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$	If aborted, then a_2^{O} else r_2^{O}	If aborted, then a_2^{O} else r_2^{O}	If aborted or recovered, then ι else $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$
R sends r_1^{R}	r_2^{R}	$[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$	If aborted, then a_2^{R} else r_2^{R}	If aborted, then a_2^{R} else r_2^{R}	If aborted or recovered, then ι else $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$

Table 1: In this table we list the choices of moves available to the trusted third party. Each row begins with a message sent by an agent to the TTP followed by the choices available to the TTP in all subsequent states. The TTP moves for the ASW, GJM and KM protocols are shown.

state of the protocol after messages m_1 and m_2 have been received. $f \downarrow X_{\mathrm{R}} = \{\mathrm{EOO}, M_2\}$ corresponds to the restriction of the valuation f to the variables of process R; all that R knows in this state is that he has received m_1 and has sent m_2. We take v_0 as the initial valuation where all variables are false. The set of variables in the refinements $\mathrm{O}' \preceq \mathrm{O}$, $\mathrm{R}' \preceq \mathrm{R}$ and $\mathrm{TTP}' \preceq \mathrm{TTP}$ are the same as those in processes O, R and TTP, respectively, and all traces begin with the initial valuation v_0. We do not model the set of channels explicitly but reason against all possible behaviors of unreliable channels. We assume that every message at least includes the name of the sender, is signed with the private key of the sender and encrypted with the public key of the recipient.

The following theorem states that the protocol model from Section 2 and the protocol synthesis model presented above are equivalent. Let $A_0 = \mathrm{O}$, $A_1 = \mathrm{R}$ and $A_2 = \mathrm{TTP}$ be the most general participants with $A = \{A_0, A_1, A_2\}$ and variables $V_0 = X_{\mathrm{O}}$ for A_0, $V_1 = X_{\mathrm{R}}$ for A_1 and $V_2 = X_{\mathrm{TTP}}$ for A_2 as defined above. It is then easy to show that,

Theorem 2 (Trace equivalence of models) *For all participant restrictions A_i' and refinements $O' \preceq O$, $R' \preceq R$ and $TTP' \preceq TTP$, such that $i \in \{0, 1, 2\}$ with $j = O$ when $i = 0$, $j = R$ when $i = 1$ and $j = TTP$ when $i = 2$, for all valuations $v \in \mathcal{F}[V_i]$, if $\Lambda_i'(v) = \Gamma_{j'}(v)$, then we have,* $\mathrm{Runs}(\{A_0', A_1', A_2'\}) = [\![O' \parallel R' \parallel TTP' \parallel Sc]\!]$.

We note that in Theorem 2, when we say all restrictions A_i' or all refinements O', R', and TTP', the most general participants are included (for example O' can be O) and hence Theorem 2 covers trace equivalence for all required cases.

5.1 Failure of Classical and Weak Co-Synthesis

In this subsection we show that classical co-synthesis fails while weak co-synthesis generates solutions that are not attack-free and are hence unacceptable. We first tackle classical co-synthesis. In order to show failure of classical co-synthesis we need to show that one of the following conditions:

1. $[\![(\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{TTP} \parallel \mathrm{Sc})]\!] \subseteq \varphi_{\mathrm{O}}$;

2. $[\![(\mathrm{O} \parallel \mathrm{R}' \parallel \mathrm{TTP} \parallel \mathrm{Sc})]\!] \subseteq \varphi_{\mathrm{R}}$;
3. $[\![(\mathrm{O} \parallel \mathrm{R} \parallel \mathrm{TTP}' \parallel \mathrm{Sc})]\!] \subseteq \varphi_{\mathrm{TTP}}$,

can be violated. We show that for all refinements R$'$ of the recipient R, that is, for every sequence of moves ending in a move chosen by R$'$, there exist moves for the other processes O, TTP and Sc, and a behavior of the the channels, to extend that sequence such that the objective φ_{R} is violated. Since R should satisfy his objective against all possible behaviors of the channels, to show failure of classical co-synthesis it suffices to fix the behavior of all channels. We assume the channels eventually deliver all messages.

Theorem 3 (Classical co-synthesis fails for R) *For all refinements $R' \preceq R$, the following assertion holds:*

$$[\![O \parallel R' \parallel TTP \parallel Sc]\!] \not\subseteq \varphi_R.$$

Proof We consider every valuation of the process variables and the set of all possible moves that can be selected by R at each valuation. This defines all possible refinements of R. Since every valuation is the result of a finite sequence of moves (messages) chosen (sent) by the agents and the TTP, it suffices to consider all possible finite sequences of messages received, ending in a message chosen by R. Let $\tau = (v_0, v_1, \ldots, v_n)$ be a finite sequence of valuations seen in a partial protocol run, where v_0 is the starting valuation. Let $\sigma = \tau \downarrow Moves = \langle a_0, a_1, \ldots, a_{n-1}\rangle$ be the corresponding sequence of n moves seen in the run. At the beginning of a protocol run, we have $\sigma = \emptyset$. In the following, on a case by case basis, we show the sequence of moves seen in a partial protocol run, ending in a move chosen by R, followed by moves for O, TTP and Sc that leads to a violation of φ_{R}.

R1: $\langle m_1, \iota\rangle$
- Whenever *Sc* schedules O, she chooses the idle action ι. Since EOO is true, as long as O does not abort the protocol but chooses to remain idle, φ_{R} is violated.
- φ_{R} and φ_{O} are violated but φ_{TTP} is satisfied.

R2: $\langle m_1, m_2\rangle$
- Sc schedules O; O sends r_1^{O} to TTP;
- Sc schedules TTP; TTP resolves the protocol for O and sends r_2^{O};
- Sc schedules O; O aborts the protocol by sending a_1^{O};
- Sc schedules TTP; TTP sends $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$ with R having no option of obtaining O's signature;
- φ_{R} and φ_{TTP} are violated but φ_{O} is satisfied.

R3: $\langle m_1, r_1^{\mathrm{R}}\rangle$
- Sc schedules TTP; TTP resolves and sends $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$;
- Sc schedules O; O sends a_1^{O} to TTP;
- Sc schedules TTP; TTP sends $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$;
- φ_{R}, φ_{TTP} and φ_{O} are violated.

R4: $\langle m_1, m_2, r_1^{\mathrm{R}}\rangle$
- Sc schedules TTP; TTP resolves and sends $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$;
- Sc schedules O; O sends a_1^{O} to TTP;
- Sc schedules TTP; TTP sends a_2^{R};
- φ_{R} and φ_{TTP} are violated but φ_{O} is satisfied.

■

It is easy to verify that the sequences in the proof are exhaustive. From the agent interface automaton shown in Figure 1 we can extract all the partial sequences of moves ending in a move of R and similarly for O. In all of the above cases, φ_{R} is violated. In all of the above cases $\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}$ is also violated. This shows that for all counter moves of O and the TTP, violation of the specification of R also violates the specification of O or the TTP. Since O and the TTP co-operate, O never sends m_3, instead choosing to use the TTP to get her non-repudiation evidence while denying R the ability to get his evidence.

The following example illustrates that given our objectives, given a reasonable TTP as defined in Section 2, weak co-synthesis yields solutions that are not attack-free and are hence unacceptable.

Example 1. **(Weak co-synthesis generates unacceptable solutions)** Consider a refinement O', R' and TTP', that generates the following sequence of messages: $\langle m_1, m_2, r_1^{\mathrm{O}}, r_2^{\mathrm{O}}, r_1^{\mathrm{R}}, r_2^{\mathrm{R}} \rangle$; the agents send m_1 and m_2 and then resolve the protocol individually. We assume that TTP' needs both m_1 and m_2 to resolve the protocol for either O or R. The trace corresponding to this sequence satisfies weak co-synthesis, but then this behavior of the TTP, that assumes co-operative agent behavior, is not attack-free. Taking $Y = \{\mathrm{R}\}$, consider the following Y-attack where R exploits the fact that a reasonable TTP responds with r_2^{R} when she receives r_1^{R}. If R sends a resolve request immediately after receiving m_1 we get the message sequence $\langle m_1, r_1^{\mathrm{R}}, r_2^{\mathrm{R}} \rangle$. In this case φ_{R} is satisfied, but φ_{O} and φ_{TTP} are violated. The only way to satisfy φ_{O} and φ_{TTP} is if O' sends r_1^{O}, which she cannot do, as she does not know the contents of m_2. This is an attack on the ASW certified mail protocol that compromises fairness for O [16]. Similarly, there exists a Y-attack for $Y = \{\mathrm{O}, \mathrm{R}\}$ as follows: after resolving the protocol, if O decides to send m_3 and R responds with m_4, we get the following message sequence: $\langle m_1, m_2, r_1^{\mathrm{O}}, r_2^{\mathrm{O}}, m_3, m_4 \rangle$. In this case, the objectives φ_{O} and φ_{R} are satisfied but the objective φ_{TTP} is violated; a reasonable TTP will only send messages in response to abort and resolve requests and thus needs r_1^{R} to satisfy φ_{TTP}. Therefore, solutions that satisfy weak co-synthesis may not be attack-free. ■

5.2 The Need for a TTP

We now provide a justification of the need for a TTP in fair non-repudiation protocols, given our synthesis objective. While this follows from [12, 21], our proof gives an alternative game-theoretic proof through synthesis. We present the following theorem which shows that if we remove the TTP, then both classical and assume-guarantee synthesis fail to synthesize a fair non-repudiation protocol.

Theorem 4 (Classical and assume-guarantee synthesis fail without the TTP) *For all refinements $O' \preceq O$, the following assertions hold:*

1. Classical co-synthesis fails: $[\![O' \parallel R \parallel Sc]\!] \not\subseteq \varphi_O$.
2. Assume-guarantee synthesis fails:
   (a) $[\![O' \parallel R \parallel Sc]\!] \not\subseteq (\varphi_R \Rightarrow \varphi_O)$ *or,*
   (b) $[\![O' \parallel R \parallel Sc]\!] \subseteq (\varphi_R \Rightarrow \varphi_O)$; $[\![R' \parallel O \parallel Sc]\!] \subseteq (\varphi_O \Rightarrow \varphi_R)$; *and* $[\![O' \parallel R' \parallel Sc]\!] \not\subseteq (\varphi_O \wedge \varphi_R)$.

Proof We note that as the TTP is not involved, AO, AR, $\mathrm{EOO}_k^{\mathrm{TTP}}$ and $\mathrm{EOR}_k^{\mathrm{TTP}}$ are always false. The agent objectives then simplify to,

$$\varphi_{\mathrm{O}} = \Diamond M_1 \wedge \Diamond \mathrm{EOR}_k^{\mathrm{R}}; \qquad \varphi_{\mathrm{R}} = \Box(\mathrm{EOO} \Rightarrow \Diamond \mathrm{EOO}_k^{\mathrm{O}}) .$$

For assertion 1, consider an arbitrary refinement $\mathrm{O}' \preceq \mathrm{O}$. We show a witness trace in $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{Sc}]\!]$ that violates φ_{O}. If O' does not send m_1 in the initial protocol state v_0, then we have a witness trace that trivially violates φ_{O} and hence $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{Sc}]\!] \not\subseteq \varphi_{\mathrm{O}}$. Assume O' sends m_1 and the channel between O and R eventually delivers all messages. Consider a partial trace ending in protocol state $\{M_1, \mathrm{EOO}, M_2, \mathrm{EOR}\}$; messages m_1 and m_2 have been received. The only choice of moves for O' in this state of the protocol are ι or m_3. If O' chooses ι, then the trace does not satisfy φ_{O} and hence $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{Sc}]\!] \not\subseteq \varphi_{\mathrm{O}}$. If O' chooses m_3 and upon receiving m_3 if R decides to stop participating in the protocol by choosing ι, then the trace satisfies φ_{R} but violates φ_{O} and hence $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{Sc}]\!] \not\subseteq \varphi_{\mathrm{O}}$.

For assertion 2, consider an arbitrary refinement $\mathrm{O}' \preceq \mathrm{O}$. If O' does not send m_1 in the initial protocol state v_0, we have a witness trace that trivially violates φ_{O} but satisfies φ_{R}. Therefore, the trace does not satisfy $\varphi_{\mathrm{R}} \Rightarrow \varphi_{\mathrm{O}}$ and $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{Sc}]\!] \not\subseteq (\varphi_{\mathrm{R}} \Rightarrow \varphi_{\mathrm{O}})$. Assume the channels eventually deliver all messages and as in the proof of assertion 1, consider a partial trace ending in protocol state $\{M_1, \mathrm{EOO}, M_2, \mathrm{EOR}\}$. To produce a witness trace we have the following cases based on the move chosen by O':

- *Case 1. O' chooses ι.* Since O' chooses ι, she does not send her signature $\mathrm{EOO}_k^{\mathrm{O}}$. Therefore, the trace does not satisfy φ_{R}. Since R sends m_4 only in response to m_3, O does not get $\mathrm{EOR}_k^{\mathrm{R}}$ from R in this case. Therefore, the trace does not satisfy φ_{O} either and hence satisfies $\varphi_{\mathrm{O}} \Rightarrow \varphi_{\mathrm{R}}$ and $\varphi_{\mathrm{R}} \Rightarrow \varphi_{\mathrm{O}}$ but does not satisfy $\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}$. This leads to, $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{O}} \Rightarrow \varphi_{\mathrm{R}})$ and $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{R}} \Rightarrow \varphi_{\mathrm{O}})$ but $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{Sc}]\!] \not\subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}})$
- *Case 2. O' chooses m_3.* Since m_3 is eventually delivered, R gets his non-repudiation evidence and the trace satisfies φ_{R}. If R now stops participating in the protocol and chooses the idle move ι instead of sending m_4, then O does not get her non-repudiation evidence and the trace does not satisfy φ_{O}. We therefore have a witness trace that does not satisfy $\varphi_{\mathrm{R}} \Rightarrow \varphi_{\mathrm{O}}$. This leads to, $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{Sc}]\!] \not\subseteq (\varphi_{\mathrm{R}} \Rightarrow \varphi_{\mathrm{O}})$

■

If the agents co-operate, then a refinement $\mathrm{O}' \preceq \mathrm{O}$ that sends m_1 and then m_3 upon receiving m_2 and similarly a refinement $\mathrm{R}' \preceq \mathrm{R}$ that sends m_2 and m_4 upon receiving m_1 and m_3 respectively, is a solution to the weak co-synthesis problem. The sequence of messages in this case is precisely $\langle m_1, m_2, m_3, m_4 \rangle$ which is the main protocol in all the fair exchange protocols we have studied. The problem arises when either O or R are dishonest and try to cheat the other agent.

5.3 Assume-guarantee Solutions are Attack-Free

In this subsection we show that assume-guarantee solutions are attack free; no coalition of participants can violate the objective of at least one of the other participants while satisfying their own objectives. Let $P' = (\mathrm{O}', \mathrm{R}', \mathrm{TTP}')$ be a tuple of refinements of the agents and the TTP. For two refinements $P' = (\mathrm{O}', \mathrm{R}', \mathrm{TTP}')$ and $P'' = (\mathrm{O}'', \mathrm{R}'', \mathrm{TTP}'')$, we write $P' \preceq P''$ if $\mathrm{O}' \preceq \mathrm{O}''$, $\mathrm{R}' \preceq \mathrm{R}''$ and $\mathrm{TTP}' \preceq \mathrm{TTP}''$. Given $P = (\mathrm{O}, \mathrm{R}, \mathrm{TTP})$, the most general behaviors of the agents and the TTP, let P_{AGS} be the set of all possible refinements $P' \preceq P$ that satisfy the conditions of assume-guarantee synthesis. For a refinement $P' = (\mathrm{O}', \mathrm{R}', \mathrm{TTP}')$ to be in P_{AGS}, we require that the refinements $\mathrm{O}' \preceq \mathrm{O}$, $\mathrm{R}' \preceq \mathrm{R}$ and $\mathrm{TTP}' \preceq \mathrm{TTP}$ satisfy the following conditions:

For all fair schedulers Sc, for all possible behaviors of the channels,

1. $[\![(\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{TTP} \parallel \mathrm{Sc})]\!] \subseteq (\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$;
2. $[\![(\mathrm{O} \parallel \mathrm{R}' \parallel \mathrm{TTP} \parallel \mathrm{Sc})]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{R}}$;

3. $[\![(\mathrm{O} \parallel \mathrm{R} \parallel \mathrm{TTP}' \parallel \mathrm{Sc})]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$;
4. $[\![(\mathrm{O}' \parallel \mathrm{R}' \parallel \mathrm{TTP}' \parallel \mathrm{Sc})]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}})$.

We now characterize the smallest restriction on the refinements $\mathrm{TTP}' \preceq \mathrm{TTP}$ that satisfy the implication condition,

$$[\![(\mathrm{O} \parallel \mathrm{R} \parallel \mathrm{TTP}' \parallel \mathrm{Sc})]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}} \ . \tag{6}$$

In order to characterize the smallest restriction on TTP' we first define the following constraints on the TTP and prove that they are both necessary and sufficient to satisfy (6).

AGS constraints on the TTP. We say that a refinement $\mathrm{TTP}' \preceq \mathrm{TTP}$ satisfies the *AGS constraints on the TTP*, if TTP' satisfies the the following constraints:

1. *Abort constraint.* If the first request received by the TTP is an abort request, then her response to that request should be $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$;
2. *Resolve constraint.* If the first request received by the TTP is a resolve request, then her response to that request should be $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$;
3. *Accountability constraint.* If the first response from the TTP is $[x, y]$, then for all subsequent abort or resolve requests her response should be in the set $\{\iota, x, y, [x, y]\}$.

We assume a reasonable TTP, as defined in Section 2; in particular she only responds to abort or resolve requests. In the following lemma, in assertion 1 we show that for all refinements $\mathrm{TTP}' \preceq \mathrm{TTP}$ that satisfy the AGS constraints on the TTP, we have TTP' is inviolable , i.e., neither agent can violate the objective φ_{TTP}, and hence satisfies the implication condition (6); in assertion (2) we show that if TTP' does not satisfy the AGS constraints on the TTP, the implication condition (6) is not satisfied.

Lemma 1 *For all refinements $TTP' \preceq TTP$, the following assertions hold:*

1. *if TTP' satisfies the AGS constraints on the TTP, then*

$$[\![O \parallel R \parallel TTP' \parallel Sc]\!] \subseteq \varphi_{TTP} \subseteq (\varphi_O \wedge \varphi_R) \Rightarrow \varphi_{TTP}.$$

2. *if TTP' does not satisfy the AGS constraints on the TTP, then*

$$[\![O \parallel R \parallel TTP' \parallel Sc]\!] \not\subseteq (\varphi_O \wedge \varphi_R) \Rightarrow \varphi_{TTP}.$$

Proof For assertion 1, consider an arbitrary $\mathrm{TTP}' \preceq \mathrm{TTP}$ that satisfies the AGS constraints on the TTP. We consider the following cases of sets of traces of $[\![\mathrm{O} \parallel \mathrm{R} \parallel \mathrm{TTP}' \parallel \mathrm{Sc}]\!]$ for the proof:

- *Case 1. Neither agent aborts nor resolves the protocol.* In these traces, since the TTP is neither sent an abort nor a resolve request, φ_{TTP} is satisfied trivially. Therefore, all these traces satisfy $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$.
- *Case 2. The first request to the TTP is an abort request.* For the set of traces where the first request to the TTP is an abort request, given TTP' satisfies the AGS constraints on the TTP, by the abort constraint, the response of the TTP to this request is $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$. For all subsequent abort or resolve requests, by the accountability constraint, the TTP responds with a move in set $\{\iota, a_2^{\mathrm{O}}, a_2^{\mathrm{R}}, [a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]\}$. This implies that both agents get the abort token and neither agent gets non-repudiation evidences. Therefore, φ_{TTP} is satisfied for all these traces and hence $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$ is also satisfied.

– *Case 3. The first request to the TTP is a resolve request.* Similar to the proof of Case 2, in the set of traces where the first request to the TTP is a resolve request, by the resolve constraint, the TTP responds to this request with move $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$. Since the response of the TTP to all subsequent abort or resolve requests is in the set $\{\iota, r_2^{\mathrm{O}}, r_2^{\mathrm{R}}, [r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]\}$, by the accountability constraint, the agents get their non-repudiation evidences and neither gets the abort token. Therefore, φ_{TTP} is satisfied for all these traces and hence $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$ is also satisfied and the result follows.

For assertion 2, consider an arbitrary $\mathrm{TTP}' \preceq \mathrm{TTP}$ that does not satisfy the AGS constraints on the TTP. We assume a reasonable TTP and consider violation of the AGS constraints on the TTP on a case by case basis. For each case we produce a witness trace that violates the implication condition $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$. We proceed as follows:

– *Case 1. The abort constraint is violated.* To produce a witness trace we consider a partial trace that ends in protocol state $\{M_1, \mathrm{ABR}^{\mathrm{O}}\}$; O requests the TTP to abort the protocol after sending message m_1 but before it is received. Since TTP' violates the abort constraint, the only choice of moves for TTP' are ι or a_2^{O}. This leads to the following cases:
  - *Case (a). TTP' chooses ι.* It is trivially the case that φ_{TTP} is violated for this trace as φ_{TTP}^1 is violated. At this stage in the protocol, there exists a behavior of O, R and the channel between O and R, where the channel delivers all messages and the agents co-operate and complete the protocol by exchanging their signatures. Therefore, $\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}$ is satisfied but φ_{TTP} is violated. Therefore, the trace does not satisfy $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$.
  - *Case (b). TTP' chooses a_2^O.* Since the channel between the agents and the TTP is resilient, O eventually receives her abort token AO. At this stage in the protocol, there exists a behavior of O, R and the channel between O and R such that the channel delivers all messages and the agents exchange their signatures, leading to the satisfaction of $\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}$ but a violation of φ_{TTP}^4 and hence φ_{TTP}. Therefore, the trace does not satisfy $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$.

– *Case 2. The resolve constraint is violated.* To produce a witness trace we consider a partial trace that ends in protocol state $\{M_1, \mathrm{EOO}, M_2, \mathrm{EOR}, \mathrm{RES}^{\mathrm{O}}\}$; O resolves the protocol after messages m_1 and m_2 have been received. Since TTP' violates the resolve constraint, the only choice of moves for TTP' are ι or r_2^{O}. An argument similar to the argument for cases 1(a) and 1(b) again leads to the satisfaction of $\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}$ but a violation of φ_{TTP}.

– *Case 3. The accountability constraint is violated.* To produce a witness trace we consider a partial trace that ends in protocol state $\{M_1, \mathrm{EOO}, M_2, \mathrm{EOR}, \mathrm{ABR}^{\mathrm{O}}, \mathrm{RES}^{\mathrm{R}}, A_2^{\mathrm{O}}, A_2^{\mathrm{R}}, \mathrm{AO}, \mathrm{AR}\}$; O aborts the protocol and R resolves the protocol after messages m_1 and m_2 have been received. The TTP receives the abort request before the resolve request and aborts the protocol by sending $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$. Since TTP' violates the accountability constraint, the only choice of moves for TTP' to the resolve request from R are r_2^{R} or $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$. The leads to the following cases:
  - *Case (a). TTP' chooses r_2^R.* This violates φ_{TTP}^4 and φ_{TTP}^5 and hence violates φ_{TTP}. At this stage in the protocol, there exists a behavior of O, R and the channel between O and R such that the agents exchange their signatures and complete the protocol thus satisfying $\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}$. Therefore, this trace does not satisfy the implication condition $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$.
  - *Case (b). TTP' chooses $[r_2^O, r_2^R]$.* This violates φ_{TTP}^4 and φ_{TTP}^5 and hence violates φ_{TTP}. An argument similar to Case 2(a) leads to a violation of $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$ for this trace.

As we have shown witness traces that do not satisfy the implication condition $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$ when TTP' violates any of the AGS constraints on the TTP, the result follows.

■

In the following theorem we show that all refinements $P' \in P_{AGS}$ are attack-free; no subset of participants can violate the objective of at least one of the other participants while satisfying their own objectives.

Theorem 5 *All refinements $P' \in P_{AGS}$ are attack-free.*

Proof We show that for all refinements $P' \in P_{AGS}$ there exists no Y-attack for all $Y \subseteq \{\mathrm{O}, \mathrm{R}, \mathrm{TTP}\}$. Let $P' = (\mathrm{O}', \mathrm{R}', \mathrm{TTP}')$ and $A = \{\mathrm{O}, \mathrm{R}, \mathrm{TTP}\}$ be the set of participants. We have the following cases:

- *Case 1.* $|Y| = 0$. In this case $Y = \emptyset$ and $(A \setminus Y)' = \{\mathrm{O}', \mathrm{R}', \mathrm{TTP}'\}$. Since $(A \setminus Y)'$ are the refinements in P' which is in P_{AGS}, by the weak co-synthesis condition, the objectives φ_{O}, φ_{R} and φ_{TTP} are satisfied. Therefore there is no Y-attack in this case.
- *Case 2.* $|Y| = 1$. We first show that there is no Y-attack for $Y = \{\mathrm{O}\}$. The case of $Y = \{\mathrm{R}\}$ is similar. By Lemma 1 (assertion 2), for all refinements $P' \in P_{AGS}$, the refinement TTP' must satisfy the AGS constraints on the TTP. This implies, by Lemma 1 (assertion 1), neither O nor R can violate φ_{TTP}. Since φ_{TTP} cannot be violated, a Y-attack in this case must generate a trace where φ_{R} is violated but φ_{O} is satisfied. But this violates the implication condition, $\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}} \Rightarrow \varphi_{\mathrm{R}}$, contradicting the assumption that $P' \in P_{AGS}$. We now show that there is no Y-attack for $Y = \{\mathrm{TTP}\}$. Since we assume the TTP is reasonable, in all traces where neither agent sends an abort nor a resolve request to the TTP, the TTP cannot violate the agent objectives. In all traces where the first request from the agents is an abort request, given a reasonable TTP, since the trace satisfies φ_{TTP}, it must be the case that the response to that request is $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$. Similarly, for resolve requests. If the first response of the TTP is $[x, y]$, then the only responses that satisfy φ_{TTP}, to all subsequent abort and resolve requests, are in the set $\{\iota, x, y, [x, y]\}$. This implies that either the agents get abort tokens or non-repudiation evidences but never both, which implies φ_{O} and φ_{R} are satisfied in all these traces. Therefore there is no Y-attack in this case as well.
- *Case 3.* $|Y| = 2$. Since $P' \in P_{AGS}$, by the implication conditions of assume-guarantee synthesis, there cannot be a Y-attack where $|Y| = 2$.
- *Case 4.* $|Y| = 3$. It is trivially the case that there is no Y-attack as $(A \setminus Y)' = \emptyset$.

Since we have shown that for all refinements $P' \in P_{AGS}$, for all $Y \subseteq A$, there is no Y-attack in P', we conclude that all refinements in P_{AGS} are attack-free. ■

We now present the following theorem that establishes conditions for any refinement in P_{AGS} to be an attack-free fair non-repudiation protocol.

Theorem 6 (Fair non-repudiation protocols) *For all refinements $P' \in P_{AGS}$, if $[\![O' \parallel R' \parallel TTP' \parallel Sc]\!] \cap (\diamond NRO \wedge \diamond NRR) \neq \emptyset$, then P' is an attack-free fair non-repudiation protocol.*

Proof Consider an arbitrary refinement $P' = (\mathrm{O}', \mathrm{R}', \mathrm{TTP}') \in P_{AGS}$. Since $P' \in P_{AGS}$, by Theorem 5, it is attack-free. Further, by the weak co-synthesis condition, we have $[\![\mathrm{O}' \parallel \mathrm{R}' \parallel \mathrm{TTP}' \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}})$ and hence by Theorem 1, we have $[\![\mathrm{O}' \parallel \mathrm{R}' \parallel \mathrm{TTP}' \parallel \mathrm{Sc}]\!] \subseteq \varphi_f$. Thus P' satisfies fairness. Using PCS, that provides the designated

verifier property, to encrypt all messages, we ensure that the protocol is abuse-free. Since $[\![\mathrm{O}' \parallel \mathrm{R}' \parallel \mathrm{TTP}' \parallel \mathrm{Sc}]\!] \cap (\Diamond\mathrm{NRO} \wedge \Diamond\mathrm{NRR}) \neq \emptyset$, the refinement P' enables an exchange of signatures and hence is an exchange protocol. Given NRO and NRR are non-repudiation evidences for R and O respectively, we conclude that P' is an attack-free fair non-repudiation protocol. ■

5.4 Analysis of Existing Fair Non-repudiation Protocols as P_{AGS} Solutions

In this subsection we analyze existing fair non-repudiation protocols and check if they are solutions to assume-guarantee synthesis. To facilitate the analysis, we first present an alternate characterization of the set P_{AGS} of assume-guarantee refinements. We then show that the KM non-repudiation protocol with offline TTP is in P_{AGS} whereas the ASW certified mail protocol and the GJM protocol are not. Finally, we present a systematic exploration of refinements leading to the KM protocol. Towards an alternate characterization of P_{AGS}, we begin by defining constraints on O, similar to the AGS constraints on the TTP that ensure satisfaction of the implication condition for O. We then define maximal and minimal refinements that satisfy all the implication conditions of assume-guarantee synthesis and introduce a *bounded idle time* requirement to ensure satisfaction of weak co-synthesis.

AGS constraints on O. Given $P = (\mathrm{O}, \mathrm{R}, \mathrm{TTP})$, the most general behaviors of the agents and the TTP, we say a refinement $P' \preceq P$ satisfies the *AGS constraints on O*, if the following conditions hold:

1. $a_1^{\mathrm{O}} \notin \Gamma_{\mathrm{O}'}(v_0)$;
2. $\mathrm{EOO}_k^{\mathrm{O}} \notin \Gamma_{\mathrm{O}'}(\{M_1, \mathrm{EOR}, \mathrm{ABR}^{\mathrm{O}}\})$; and
3. $a_1^{\mathrm{O}} \notin \Gamma_{\mathrm{O}'}(\{M_1, \mathrm{EOR}, M_3\})$.

In the Appendix, we show that these constraints are both necessary and sufficient restrictions on the moves of O that satisfy the implication condition $(\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$ of assume-guarantee synthesis. We also show that all refinements $\mathrm{R}' \preceq \mathrm{R}$ satisfy the implication condition $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{R}}$ of assume-guarantee synthesis.

The maximal refinement P^*. We define the maximal refinement $\mathrm{P}^* = (\mathrm{O}^*, \mathrm{R}^*, \mathrm{TTP}^*)$ as follows:

1. $\mathrm{O}^* \preceq \mathrm{O}$ satisfies the AGS constraints on O and for all O' that satisfy the constraints, we have $O' \preceq \mathrm{O}^*$;
2. $\mathrm{R}^* = \mathrm{R}$; and
3. $\mathrm{TTP}^* \preceq \mathrm{TTP}$ satisfies the AGS constraints on the TTP and for all TTP' that satisfy the constraints, we have $\mathrm{TTP}' \preceq \mathrm{TTP}^*$.

We show in the Appendix the correspondence between P^* and the smallest restriction on the moves of O and the TTP so that P^* is a witness to P_{AGS}. While there are restrictions on O and the TTP, there are no restrictions on R.

The minimal refinement P_*. We present the smallest refinement $\mathrm{P}_* = (\mathrm{O}_*, \mathrm{R}_*, \mathrm{TTP}_*)$ in P_{AGS}, as the largest restriction on the moves of O, R and the TTP, as follows:

1. $\mathrm{P}_* \preceq \mathrm{P}^*$;
2. $\mathit{Moves}_{\mathrm{O}_*} = \{m_1, a_1^{\mathrm{O}}\}$;
3. $\mathit{Moves}_{\mathrm{R}_*} = \{\iota\}$;
4. O_* satisfies the AGS constraints on O; and
5. TTP_* satisfies the AGS constraints on the TTP.

Protocol 1: THE KM, ASW AND GJM MAIN PROTOCOL

O sends m_1 to R;
R sends m_2 to O;
if *(R does not send m_2 on time)* **then**
  O sends a_1^{O} to the TTP;
else
  O sends m_3 to R;
  if *(O does not send m_3 on time)* **then**
    R sends r_1^{R} to the TTP;
  else
    R sends m_4 to O;
    if *(R does not send m_4 on time)* **then**
      O sends r_1^{O} to the TTP;

If $m_1 \notin \mathit{Moves}_{\mathrm{O}_*}$, then φ_{O} cannot be satisfied as O_* does have the ability to initiate a protocol instance. If $a_1^{\mathrm{O}} \notin \mathit{Moves}_{\mathrm{O}_*}$, then φ_{O} cannot be satisfied whether or not m_1 is delivered, as R_* has no choice of moves other than ι. If O_* does not satisfy the AGS constraints on O and sends a_1^{O} in the initial state of the protocol v_0, then the resulting trace trivially violates φ_{O} while satisfying $\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}$.

The bounded idle time requirement. We say that a refinement P' satisfies *bounded idle time* if O and the TTP in P' choose the idle move ι, when scheduled by Sc, at most b times for a finite $b \in \mathbb{N}$. We prove that satisfaction of the bounded idle time requirement is both necessary and sufficient to ensure satisfaction of the weak co-synthesis condition of assume-guarantee synthesis, for all refinements that satisfy the AGS constraints on the TTP and the AGS constraints on O, in the Appendix.

Alternate characterization of P_{AGS}. We now use P_* and P^* to provide an alternate characterization of the set P_{AGS}. We first define the following set of refinements $\overline{\mathrm{P}}$:

$$\begin{aligned}\overline{\mathrm{P}} = \{P' = (\mathrm{O}', \mathrm{R}', \mathrm{TTP}') \mid\ & P' \text{ satisfies bounded idle time}; \mathrm{P}_* \preceq P' \preceq \mathrm{P}^*; \\ & \mathrm{TTP}' \text{ satisfies the AGS constraints on the TTP}\}\,.\end{aligned}$$

The following lemma states that the set $\overline{\mathrm{P}}$ and the set P_{AGS} coincide. We present the lemma here and prove it in the Appendix.

Lemma 2 (Alternate characterization of P_{AGS}) *We have $\overline{P} = P_{AGS}$.*

The KM non-repudiation protocol. The KM protocol, like the ASW and GJM protocols consists of a main protocol, an abort subprotocol and a resolve subprotocol. The main protocol is the same as in the ASW and GJM protocols and is defined in terms of messages in Protocol 1. The abort subprotocol and the resolve subprotocol are defined in Table 1. Let $P_{KM} = (\mathrm{O}_{KM}, \mathrm{R}_{KM}, \mathrm{TTP}_{KM})$ correspond to the agent and TTP refinements in the KM protocol. Since O does not abort the protocol in state v_0 and in state $\{M_1, \mathrm{EOR}, M_3\}$ in O_{KM}, it follows that $\mathrm{O}_* \preceq \mathrm{O}_{KM} \preceq \mathrm{O}^*$. It is easy to verify that $\mathrm{R}_* \preceq \mathrm{R}_{KM} \preceq \mathrm{R}^*$ and $\mathrm{TTP}_* \preceq \mathrm{TTP}_{KM} \preceq \mathrm{TTP}^*$. Moreover, TTP_{KM} satisfies the AGS constraints on the TTP and P_{KM} satisfies bounded idle time. Therefore $P_{KM} \in \overline{\mathrm{P}}$ and hence by Lemma 2, $P_{KM} \in P_{AGS}$.

The ASW certified mail protocol. The ASW certified mail protocol differs from the KM protocol in its abort and resolve sequences. To define the abort protocol, the TTP needs a move req^O that can be used to request O to resolve a protocol instance if R has already resolved it. The abort and resolve subprotocols are defined in Table 1. Let $P_{ASW} = (\mathrm{O}_{ASW}, \mathrm{R}_{ASW}, \mathrm{TTP}_{ASW})$ correspond to the agent and TTP refinements in the ASW certified mail protocol. Since TTP_{ASW} neither has move $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$ nor $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$, TTP_{ASW} does not satisfy the AGS constraints on the TTP and hence by Lemma 1 (assertion 2), we have $P_{ASW} \notin P_{AGS}$. Moreover, the ASW certified mail protocol is not attack-free as shown by the following attacks [16]: Consider a behavior of the channels that deliver all messages and the sequence of messages $\langle m_1, r_1^{\mathrm{R}}, r_2^{\mathrm{R}}, a_1^{\mathrm{O}}, req^O \rangle$. This is a valid sequence in the ASW protocol. In this sequence a malicious R decides to resolve the protocol after receiving m_1 and thus succeeds in getting $\mathrm{EOO}_k^{\mathrm{TTP}}$. When O_{ASW} attempts to abort the protocol, TTP_{ASW} expects her to resolve the protocol as R has already resolved it, but O_{ASW} cannot do so as she does not have m_2. Therefore, φ_{O} is violated; O_{ASW} cannot abort or resolve the protocol, neither can she get R's signature. Consider the sequence of messages $\langle m_1, m_2, r_1^{\mathrm{O}}, r_2^{\mathrm{O}}, a_1^{\mathrm{O}}, a_2^{\mathrm{O}} \rangle$. This is an attack that compromises fairness for R; in the words of [16] the protocol designers did not foresee that O could resolve the protocol and then abort it. This violates φ_{R} and TTP accountability, violating φ_{TTP}, while satisfying φ_{O}.

The GJM protocol. The GJM protocol differs in the abort and resolve sequences as shown in Table 1. Garay et al., introduced the notion of abuse-freeness and invented *private contract signatures or PCS*, a cryptographic primitive that ensures abuse-freeness and optionally TTP accountability [13]. Further, the GJM protocol is faithful to the informal definition of fairness in that, when a protocol instance is aborted, neither agent gets partial information that can be used to negotiate a contract with a third party. This is ensured by the use of PCS which provides the *designated verifier property*; only R can verify the authenticity of a message signed by O and vice versa. The use of PCS in addition to the fixes to the original protocol proposed in [28] ensure that the protocol is free from replay attacks, is fair and abuse-free. Let $P_{GJM} = (\mathrm{O}_{GJM}, \mathrm{R}_{GJM}, \mathrm{TTP}_{GJM})$ correspond to the agent and TTP refinements in the GJM protocol. Since TTP_{GJM} neither has move $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$ nor $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$, TTP_{GJM} does not satisfy the AGS constraints on the TTP and hence by Lemma 1 (assertion 2), we have $P_{GJM} \notin P_{AGS}$. P_{GJM} does not provide TTP inviolability and is not attack-free by our definition. Consider the message sequence $g = \langle m_1, m_2, m_3, r_1^{\mathrm{O}}, r_2^{\mathrm{O}} \rangle$; agent R does not send his final signature but goes idle and stops participating in the protocol after receiving O's signature. O_{GJM} resolves the protocol by sending r_1^{O} and gets $\mathrm{EOR}_k^{\mathrm{TTP}}$. In this case, while the objectives of O and R are satisfied, the TTP cannot satisfy φ_{TTP} unless R_{GJM} co-operates and sends a resolve request r_1^{R} after having satisfied his objective, which he may never do; it is rather unrealistic to expect that he will. Precisely, $g \in [\![\mathrm{O} \parallel \mathrm{R} \parallel \mathrm{TTP}_{GJM} \parallel \mathrm{Sc}]\!]$ and $g \notin (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$.

Theorem 7 *The refinement corresponding to the KM non-repudiation protocol is in P_{AGS} and the refinements corresponding to the ASW certified mail protocol and the GJM protocol are not in P_{AGS}.*

Computation. We can obtain the solution of assume-guarantee synthesis by solving graph games with *secure equilibria* [10]. In fact, the refinements that satisfy assume-guarantee synthesis precisely correspond to secure equilibrium strategies of players in the game. This result was presented in [9]. All the objectives we consider in this paper are boolean combinations of Büchi ($\Box\Diamond$) and co-Büchi ($\Diamond\Box$) objectives. It follows from [9] that secure equilibria with combinations of Büchi and co-Büchi objectives can be solved in polynomial time. This

Delivered message sequences	Moves for $\mathbf{O}_1$ and $\mathbf{R}_1$				
	Choices for $\mathbf{O}_1$		Choices for $\mathbf{R}_1$		
$\langle\rangle$	m_1	m_1	ι	ι	ι
$\langle \mathbf{m_1} \rangle$	ι	a_1^{O}	ι	m_2	either ι or m_2
$\langle \mathbf{m_1}, \mathbf{m_2} \rangle$	a_1^{O}	a_1^{O}	ι	ι	ι
$\langle \mathbf{m_1}, \mathbf{m_2}, \mathbf{m_3} \rangle$	$\emptyset$	$\emptyset$	ι	m_4	ι

Table 2: The moves that satisfy the objectives of assume-guarantee synthesis for O_1 and R_1 are shown in this table at relevant protocol states represented by message sequences, when the agents have no ability to resolve the protocol.

gives us a polynomial time algorithm for the assume-guarantee synthesis of fair exchange protocols.

From P_{AGS} to P_{KM}. We now first present a systematic exploration of the refinements of $P = (\mathrm{O}, \mathrm{R}, \mathrm{TTP})$, the most general behavior of the agents and the TTP, leading to the KM protocol. We consider the following refinements, that we assume satisfy bounded idle time and the AGS constraints on the TTP, and study their properties:

1. $\mathrm{P}_* = (\mathrm{O}_*, \mathrm{R}_*, \mathrm{TTP}_*)$; the minimal refinement.
2. $P_1 = (\mathrm{O}_1, \mathrm{R}_1, \mathrm{TTP}_1)$ with $\mathit{Moves}_{\mathrm{O}_1} = \mathit{Moves}_{\mathrm{O}_*} \cup \{\iota, m_3\}$, $\mathit{Moves}_{\mathrm{R}_1} = \mathit{Moves}_{\mathrm{R}_*} \cup \{m_2, m_4\}$ and $\mathrm{TTP}_1 = \mathrm{TTP}^*$.
3. $P_2 = (\mathrm{O}_2, \mathrm{R}_2, \mathrm{TTP}_2)$ with $\mathit{Moves}_{\mathrm{O}_2} = \mathit{Moves}_{\mathrm{O}_1} \cup \{r_1^{\mathrm{O}}\}$, $\mathit{Moves}_{\mathrm{R}_2} = \mathit{Moves}_{\mathrm{R}_1}$ and $\mathrm{TTP}_2 = \mathrm{TTP}^*$.
4. $P_3 = (\mathrm{O}_3, \mathrm{R}_3, \mathrm{TTP}_3)$ with $\mathit{Moves}_{\mathrm{O}_3} = \mathit{Moves}_{\mathrm{O}_2} \setminus \{a_1^{\mathrm{O}}\}$, $\mathit{Moves}_{\mathrm{R}_3} = \mathit{Moves}_{\mathrm{R}_1} \cup \{r_1^{\mathrm{R}}\}$ and $\mathrm{TTP}_3 = \mathrm{TTP}^*$.
5. $\mathrm{P}^* = (\mathrm{O}^*, \mathrm{R}^*, \mathrm{TTP}^*)$; the maximal refinement.

Analysis of the refinement P_*. It is easy to check that while $\mathrm{P}_* \in P_{AGS}$, it always ends aborted as a_1^{O} is the only choice of moves for O_* after m_1 is sent. It is not an exchange protocol as it does not enable an exchange of signatures.

Analysis of the refinement P_1. In this case, the agents do not have the ability to resolve the protocol. The objectives of the agent and the TTP then reduce to,

$$\varphi_{\mathrm{O}} = \Diamond M_1 \wedge \Box(\Diamond \mathrm{EOR}_k^{\mathrm{R}} \vee (\Diamond \mathrm{AO} \wedge \Box\neg \mathrm{EOO}_k^{\mathrm{O}})),$$
$$\varphi_{\mathrm{R}} = \Box(\mathrm{EOO} \Rightarrow (\Diamond \mathrm{EOO}_k^{\mathrm{O}} \vee (\Diamond \mathrm{AR} \wedge \Box\neg \mathrm{EOR}_k^{\mathrm{R}})),$$
$$\varphi_{\mathrm{TTP}} = \Box(\mathrm{ABR} \Rightarrow (\Diamond \mathrm{AO} \vee \Diamond \mathrm{AR})) \wedge \Box(\mathrm{AO} \Rightarrow \Diamond \mathrm{AR}) \wedge \Box(\mathrm{AR} \Rightarrow \Diamond \mathrm{AO}) .$$

The agent moves that extend partial protocol runs such that the implication conditions of assume-guarantee synthesis are satisfied in all resulting traces is shown in Table 2. Each row in the table corresponds to a protocol state and the moves available to O_1 and R_1 at that state, such that the implication conditions of assume-guarantee synthesis are satisfied in all resulting traces. For example, in the row corresponding to $\langle m_1 \rangle$, we have two move choices for O_1, one that selects ι and the other that selects a_1^{O}; O_1 can choose to wait for R to send m_2 or choose a_1^{O}. A similar interpretation is attached to the moves of R_1. We have $\mathrm{P}_* \preceq P_1 \preceq \mathrm{P}^*$. As P_1 satisfies bounded idle time and the AGS constraints on the TTP, $P_1 \in \overline{\mathrm{P}}$ and hence, by Lemma 2, $P_1 \in P_{AGS}$. The refinement P_1, while attack-free, is not a fair non-repudiation protocol as it does not enable an exchange of non-repudiation evidences. The protocol always ends up aborted as a_1^{O} is the only move that satisfies φ_{O} for

Delivered message sequences	Moves for O_3 and R_3				
	Choices for O_3		Choices for R_3		
$\langle\rangle$	m_1	m_1	ι	ι	ι
$\langle \mathbf{m_1} \rangle$	ι	ι	m_2	r_1^{R}	either m_2 or r_1^{R}
$\langle \mathbf{m_1}, \mathbf{m_2} \rangle$	m_3	r_1^{O}	ι	r_1^{R}	either ι or r_1^{R}
$\langle \mathbf{m_1}, \mathbf{m_2}, \mathbf{m_3} \rangle$	ι	r_1^{O}	ι	m_4	r_1^{R}

Table 3: The moves that satisfy the objectives of assume-guarantee synthesis for O_3 and R_3 are shown in this table at relevant protocol states represented by message sequences, when the agents have no ability to abort the protocol.

O in state $\{M_1, \mathrm{EOO}\}$ against all behaviors of R and the TTP; once O_1 sends her signature in m_3, there is no move available to O_1 such that satisfaction of $\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}$ is guaranteed to satisfy φ_{O}, as R may decide to stop participating in the protocol.

Analysis of the refinement $\mathbf{P}_2$. In this case, R has no ability to resolve the protocol. It is easy to verify that $\mathrm{P}_* \preceq P_2 \preceq \mathrm{P}^*$. Therefore, $P_2 \in \overline{\mathrm{P}}$ and hence, by Lemma 2, $P_2 \in P_{AGS}$. This protocol is a fair non-repudiation protocol that satisfies fairness, balance and timeliness. If O does not send m_3, then R_2 has no choice of moves. But since P_2 satisfies bounded idle time, O_2 will eventually either abort or resolve the protocol. As TTP_2 satisfies the AGS constraints on the TTP, either both agents get abort tokens or they get their respective non-repudiation evidences eventually.

Analysis of the refinement $\mathbf{P}_3$. Since O has no ability to abort the protocol, while both agents have the ability to resolve it, the predicates AO and AR are always false. The agent and TTP objectives then reduce to,

$$\varphi_{\mathrm{O}} = \Diamond M_1 \wedge \Box(\Diamond \mathrm{EOR}_k^{\mathrm{R}} \vee \Diamond \mathrm{EOR}_k^{\mathrm{TTP}}),$$
$$\varphi_{\mathrm{R}} = \Box(\mathrm{EOO} \Rightarrow (\Diamond \mathrm{EOO}_k^{\mathrm{O}} \vee \Diamond \mathrm{EOO}_k^{\mathrm{TTP}})),$$
$$\varphi_{\mathrm{TTP}} = \Box(\mathrm{RES} \Rightarrow (\Diamond \mathrm{EOO}_k^{\mathrm{TTP}} \vee \Diamond \mathrm{EOR}_k^{\mathrm{TTP}})) \wedge \Box(\mathrm{EOO}_k^{\mathrm{TTP}} \Rightarrow \Diamond \mathrm{EOR}_k^{\mathrm{TTP}}) \wedge \Box(\mathrm{EOR}_k^{\mathrm{TTP}} \Rightarrow \Diamond \mathrm{EOO}_k^{\mathrm{TTP}}) .$$

The moves of the agents that satisfy the objectives of assume-guarantee synthesis at select protocol valuations represented by message sequences are shown in Table 3. It is easy to verify that as $\mathrm{P}_* \not\preceq P_3 \preceq \mathrm{P}^*$, $P_3 \notin \overline{\mathrm{P}}$ and hence by Lemma 2, $P_3 \notin P_{AGS}$. Since TTP_3 satisfies the AGS constraints on the TTP, P_3 is a fair non-repudiation protocol similar to the ZG optimistic non-repudiation protocol, but it does not satisfy timeliness [14] as O does have the ability to abort the protocol. If message m_1 is not delivered, then O has no choice of moves to satisfy φ_{O}, while $\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}$ are satisfied trivially. Balance does not apply in this case as there are no abort moves.

Analysis of the refinement $\mathbf{P}^*$. In the maximal refinement $\mathrm{P}^* = (\mathrm{O}^*, \mathrm{R}^*, \mathrm{TTP}^*)$, since TTP^* satisfies the AGS constraints on the TTP, if her first response to an abort or resolve request is $[x, y]$, she can choose any move in $\{\iota, x, y, [x, y]\}$ for all subsequent abort or resolve requests. Consider a refinement $P_{KM} = (\mathrm{O}_{KM}, \mathrm{R}_{KM}, \mathrm{TTP}_{KM}) \preceq \mathrm{P}^*$, where O_{KM} and R_{KM} correspond to O^* and R^* and $\mathrm{TTP}_{KM} \preceq \mathrm{TTP}^*$ such that TTP_{KM} goes idle after her first response to an abort or resolve request. P_{KM} is then the KM protocol. We remark that given the choices of moves for the TTP after her first response as suggested by assume-guarantee synthesis, choosing ι satisfies the informal notion of efficiency. This refinement ensures fairness, balance and timeliness.

Protocol 2: Main Protocol of our Symmetric Non-repudiation Protocol

```
O sends m_1 to R;
if (R does not want to participate) then
    R sends a_1^R to the TTP;
else
    R sends m_2 to O;
    if (R does not send m_2 on time) then
        O sends a_1^O to the TTP;
    else
        O sends m_3 to R;
        if (O does not send m_3 on time) then
            if (R does not want to participate) then
                R sends a_1^R to the TTP;
            else
                R sends r_1^R to the TTP;
        else
            R sends m_4 to O;
            if (R does not send m_4 on time) then
                O sends r_1^O to the TTP;
```

6 A Symmetric Fair Non-Repudiation Protocol

In the KM, ASW and GJM protocols, R cannot abort the protocol. While the ability of O to abort the protocol after sending m_1 is required in the event m_1 is not delivered or R does not send m_2, it can be used to abort the protocol even if all channels are resilient or if O decides not to sign the contract after receiving m_2. The protocols give O the ability to postpone abort decisions but deny R a similar ability. While this does not violate fairness or abuse-freeness as per prevailing definitions, it is not equitable to both agents. If R does not want to participate in a protocol instance, then the only choice of moves for R is ι and not m_2; O will then eventually abort the protocol. Once m_2 has been sent, if R decides not to participate in the protocol and not be held responsible for signing the contract, he has no choice of moves. If he decides to ignore m_3, then O will resolve the protocol resulting in non-repudiation evidences being issued to O, using which she can claim R is obligated by the contract.

In this section we present a symmetric fair non-repudiation protocol that gives R the ability to abort the protocol, assuming that the channels between the agents and the TTP are operational. If we enhance the ability of R by including an abort move a_1^{R} without enhancing O and the TTP, then assume-guarantee synthesis fails. By enhancing both O and the TTP, using assume-guarantee analysis, we design a new fair non-repudiation protocol that (a) has no Y-attack for all $Y \subseteq \{\mathrm{O}, \mathrm{R}\}$; and (b) that provides R the ability to abort. In the following, we show that if we fix the behavior of the TTP, ensuring TTP inviolability, then the protocol is attack-free.

Consider the following refinement $\mathrm{P}_s = (\mathrm{O}_s, \mathrm{R}_s, \mathrm{TTP}_s)$ with $\mathrm{P}^* \preceq \mathrm{P}_s$ defined as follows:

$Moves_{\mathrm{O}_s} = Moves_{\mathrm{O}^*} \cup \{res^{\mathrm{O}}\}$;
$Moves_{\mathrm{R}_s} = Moves_{\mathrm{R}^*} \cup \{a_1^{\mathrm{R}}\}$; and
$Moves_{\mathrm{TTP}_s} = Moves_{\mathrm{TTP}^*} \cup \{req^{\mathrm{O}}\}$.

Protocol 3: ABORT SUBPROTOCOL. $X \in \{\mathrm{O}, \mathrm{R}\}$

X sends a_1^X to TTP;
if *(the protocol has been aborted or resolved)* **then**
    TTP goes idle;
else
    if *($X = R$)* **then**
        TTP sends req^{O} to O;
        if *(O sends res^{O} on time)* **then**
            TTP marks this protocol instance as resolved in its persistent DB;
            TTP sends $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$ to O and R;
        else
            TTP marks this protocol instance as aborted in its persistent DB;
            TTP sends $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$ to O and R;
    else
        TTP marks this protocol instance as aborted in its persistent DB;
        TTP sends $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$ to O and R;

The move req^{O} may be sent by TTP_s only after receiving an abort request from R. The move res^{O} may be sent by O_s only after receiving req^{O}. We present the main protocol and the abort subprotocol for our symmetric fair non-repudiation protocol in Protocol 2 and Protocol 3; the resolve subprotocol is identical to the one in the KM protocol.

To facilitate the assume-guarantee analysis of P_s, we present the following *enhanced AGS constraints on the TTP* that is both necessary and sufficient to ensure TTP inviolability (neither agent can violate φ_{TTP}):

1. *Abort constraint.* If the first request received by the TTP is a_1^{O}, then her response to that request should be $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$; If the first request received by the TTP is a_1^{R}, then her response to that request should be req^{O};
2. *Resolve constraint.* If the first request received by the TTP is a resolve request, then her response to that request should be $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$; If the TTP receives res^{O} in response to req^{O} within bounded idle time, then her response should be $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$, otherwise it should be $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$.
3. *Accountability constraint.* If the first response from the TTP is $[x, y]$ or the first response from the TTP is req^{O} and the next response is $[x, y]$, then for all subsequent abort or resolve requests her response should be in the set $\{\iota, x, y, [x, y]\}$.

The enhanced AGS constraints on the TTP are required both to satisfy the implication condition $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$ and the condition for weak co-synthesis, $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}})$. Since TTP_s waits for a bounded number of turns before sending abort tokens to both agents after sending req^{O}, we require that (a) the channels between the agents and the TTP are operational, and (b) the time taken to deliver messages req^{O} and res^{O} be subsumed by the bound on idle time chosen by the TTP between sending req^{O} and abort tokens. As there is no bound on the time taken to deliver messages on resilient channels, the above AGS constraints on the TTP cannot be enforced without operational channels. Consider a partial trace that ends in protocol state $\{M_1, \mathrm{EOO}, M_2, \mathrm{EOR}, M_3\}$; messages m_1 and m_2 have been received and m_3 has been sent. If R now aborts the protocol and the TTP sends req^{O} to O, then resilient channels can delay delivering either req^{O} or res^{O} sufficiently for the TTP to abort the protocol. In this case if m_3 is eventually delivered, φ_{O} is violated whereas $\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}$ is satisfied.

In the following lemma we show that in P_s, O cannot violate φ_{R} while satisfying φ_{O}, R cannot violate φ_{O} while satisfying φ_{R} and O and R cannot violate φ_{TTP} while satisfying their objectives. That is, in the refinement P_s we have $[\![\mathrm{O} \parallel \mathrm{R} \parallel \mathrm{TTP}_s \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$, and $[\![\mathrm{O} \parallel \mathrm{R}_s \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{R}}$. However, it is not the case that $[\![\mathrm{O}_s \parallel \mathrm{R} \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$. But if the TTP is fixed then the implication condition holds, i.e., $[\![\mathrm{O}_s \parallel \mathrm{R} \parallel \mathrm{TTP}_s \parallel \mathrm{Sc}]\!] \subseteq \varphi_{\mathrm{R}} \Rightarrow \varphi_{\mathrm{O}} \subseteq (\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$. It follows that under the assumption that the TTP does not change her behavior, while satisfying her objective, the symmetric protocol is attack-free. We present the following lemma and prove it in the Appendix.

Lemma 3 *For the refinement $P_s = (O_s, R_s, TTP_s)$, if the channels between the agents and the TTP are operational, then there exists no Y-attack for all $Y \subseteq \{O, R\}$.*

The assumption that the bound on idle time of the TTP between sending req^{O} and abort tokens subsume the time taken for the delivery of messages req^{O} and res^{O} can easily be enforced before the beginning of a protocol; O agrees to participate in the protocol with a given TTP, only if the bound chosen by the TTP is satisfactory. We point out that in state $\{\mathrm{EOO}, M_2\}$, if R sends an abort request, he still needs O's co-operation to abort the protocol. Since she has m_2, she can launch recovery if she so desires by composing res^{O} when she receives req^{O}. But this is identical to the ability of O in aborting the protocol after she sends m_1. R can resolve the protocol as soon as he receives m_1 and thus hold O as a signatory to the contract even if she decided to abort the protocol after sending m_1. The protocol is therefore symmetrical to both O and R. In addition, we claim that this version of the protocol provides better quality of service in terms of timeliness; O does not have to wait after sending m_1 for R to send m_2, in protocol instances where R has no desire to sign the contract. The following theorem states that if the TTP does not change her behavior, then the refinement P_s is an attack-free fair non-repudiation protocol. The proof is in the Appendix.

Theorem 8 (Symmetric attack-free protocol) *Given the channels between the agents and the TTP are operational and the TTP does not deviate from satisfying the enhanced AGS constraints on the TTP, the refinement $P_s = (O_s, R_s, TTP_s)$ is an attack-free fair non-repudiation protocol.*

From P_{AGS} to P_s. We can systematically analyze refinements leading to P_s. Similar to the case of synthesizing the KM non-repudiation protocol, we now present the steps that explore refinements leading to P_s. We assume the TTP satisfies the AGS constraints on the TTP and all refinements satisfy bounded idle time. The analyzed refinements are as follows:

1. $\mathrm{P}_* = (\mathrm{O}_*, \mathrm{R}_*, \mathrm{TTP}_*)$; the minimal refinement.
2. $P_1 = (\mathrm{O}_1, \mathrm{R}_1, \mathrm{TTP}_1)$ with $\mathit{Moves}_{\mathrm{O}_1} = \mathit{Moves}_{\mathrm{O}_*} \cup \{\iota, m_3\}$, $\mathit{Moves}_{\mathrm{R}_1} = \mathit{Moves}_{\mathrm{R}_*} \cup \{m_2, m_4\}$ and $\mathrm{TTP}_1 = \mathrm{TTP}^*$.
3. $P_2 = (\mathrm{O}_2, \mathrm{R}_2, \mathrm{TTP}_2)$ with $\mathit{Moves}_{\mathrm{O}_2} = \mathit{Moves}_{\mathrm{O}_1} \cup \{r_1^{\mathrm{O}}\}$, $\mathit{Moves}_{\mathrm{R}_2} = \mathit{Moves}_{\mathrm{R}_1}$ and $\mathrm{TTP}_2 = \mathrm{TTP}^*$.
4. $P_3 = (\mathrm{O}_3, \mathrm{R}_3, \mathrm{TTP}_3)$ with $\mathit{Moves}_{\mathrm{O}_3} = \mathit{Moves}_{\mathrm{O}_2} \setminus \{a_1^{\mathrm{O}}\}$, $\mathit{Moves}_{\mathrm{R}_3} = \mathit{Moves}_{\mathrm{R}_1} \cup \{r_1^{\mathrm{R}}\}$ and $\mathrm{TTP}_3 = \mathrm{TTP}^*$.
5. $\mathrm{P}^* = (\mathrm{O}^*, \mathrm{R}^*, \mathrm{TTP}^*)$; the maximal refinement.
6. $\mathrm{P}_s = (\mathrm{O}_s, \mathrm{R}_s, \mathrm{TTP}_s)$ with $\mathit{Moves}_{\mathrm{O}_s} = \mathit{Moves}_{\mathrm{O}^*} \cup \{res^{\mathrm{O}}\}$, $\mathit{Moves}_{\mathrm{R}_s} = \mathit{Moves}_{\mathrm{R}^*} \cup \{a_1^{\mathrm{R}}\}$ and $\mathit{Moves}_{\mathrm{TTP}_s} = \mathit{Moves}_{\mathrm{TTP}^*} \cup \{req^{\mathrm{O}}\}$.

Implementation. We have implemented a prototype for assume-guarantee synthesis of fair non-repudiation protocols. Our implementation considers triples of refinements $O' \preceq \mathrm{O}$, $R' \preceq \mathrm{R}$, and $\mathrm{TTP}' \preceq \mathrm{TTP}$ and then explores all possible message sequences given these participant refinements. We implemented a scheduler that backtracks and systematically schedules all participants at all protocol states. Using the scheduler, given a subset of participant refinements, with all other participants being most general, the implementation explores all possible traces and checks if each trace satisfies the required AGS conditions. Note that in checking the satisfaction of the AGS conditions, for the implication conditions we need to consider the most general participants against each of the refinements O', R' and TTP'. The checking of the implication conditions is achieved by solving secure equilibrium on graph games with lexicographic objectives. Our implementation generates all possible AGS solutions. The analysis of the AGS solutions generated by our implementation was key in obtaining the symmetric protocol; using a procedure similar to obtaining P_{KM} from P_{AGS}.

7 Conclusion

In this work we introduce and demonstrate the effectiveness of assume-guarantee synthesis in synthesizing fair exchange protocols. Our main goal is to introduce a general assume-guarantee synthesis framework that can be used with a variety of objectives; we considered a TTP objective that treats the agents symmetrically, but the framework can be used with possibly weaker TTP objectives that treat agents asymmetrically. Using assume-guarantee analysis we have obtained a new symmetric protocol that is attack-free, given the channels to the TTP are operational. While the need for operational channels may be considered impractical, we remark that it is this flexible framework that could automatically generate such protocols of theoretical interest in the first place. For future work we will study the application of assume-guarantee synthesis to other security protocols.

References

1. Almudena Alcaide, Juan M. Estévez-Tapiador, Julio César Hernández Castro, and Arturo Ribagorda. Nature-inspired synthesis of rational protocols. In *PPSN*, pages 981–990, 2008.
2. Rajeev Alur, Thomas A. Henzinger, Freddy Y. C. Mang, Shaz Qadeer, Sriram K. Rajamani, and Serdar Tasiran. Mocha: Modularity in model checking. In *CAV*, pages 521–525, 1998.
3. Alessandro Armando, Roberto Carbone, and Luca Compagna. Ltl model checking for security protocols. In *CSF*, pages 385–396, 2007.
4. N. Asokan, Victor Shoup, and Michael Waidner. Asynchronous protocols for optimistic fair exchange. In *IEEE S&P*, pages 86–99, 1998.
5. Michael Burrows, Martín Abadi, and Roger M. Needham. A logic of authentication. *ACM Trans. Comput. Syst.*, 8(1):18–36, 1990.
6. Rohit Chadha, Max I. Kanovich, and Andre Scedrov. Inductive methods and contract-signing protocols. In *CCS*, pages 176–185. ACM, 2001.
7. Rohit Chadha, Steve Kremer, and Andre Scedrov. Formal analysis of multiparty contract signing. *J. Autom. Reasoning*, 36(1-2):39–83, 2006.
8. Rohit Chadha, John C. Mitchell, Andre Scedrov, and Vitaly Shmatikov. Contract signing, optimism, and advantage. In *CONCUR*, pages 361–377, 2003.
9. Krishnendu Chatterjee and Thomas A. Henzinger. Assume-guarantee synthesis. In *TACAS*, pages 261–275, 2007.
10. Krishnendu Chatterjee, Thomas A. Henzinger, and Marcin Jurdzinski. Games with secure equilibria. *Theor. Comput. Sci.*, 365(1-2):67–82, 2006.

11. Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs, Workshop*, pages 52–71. Springer-Verlag, 1982.
12. S. Even and Y. Yacobi. Relations among public key signature systems, technical report 175. Technical report, Technion, Haifa, Israel, 1980.
13. Juan A. Garay, Markus Jakobsson, and Philip D. MacKenzie. Abuse-free optimistic contract signing. In *CRYPTO*, pages 449–466, 1999.
14. Steve Kremer, Olivier Markowitch, and Jianying Zhou. An intensive survey of fair non-repudiation protocols. *CC*, 25(17):1606–1621, 2002.
15. Steve Kremer and Jean-François Raskin. Game analysis of abuse-free contract signing. In *CSFW*, pages 206–220. IEEE, 2002.
16. Steve Kremer and Jean-François Raskin. A game-based verification of non-repudiation and fair exchange protocols. *JCS*, 11(3):399–430, 2003.
17. Panagiotis Louridas. Some guidelines for non-repudiation protocols. *SIGCOMM CC*, 30(5):29–38, 2000.
18. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, New York, 1991.
19. Olivier Markowitch, Dieter Gollmann, and Steve Kremer. On fairness in exchange protocols. In *ICISC*, pages 451–464, 2002.
20. Olivier Markowitch and Steve Kremer. An optimistic non-repudiation protocol with transparent trusted third party. In *ISC*, pages 363–378, 2001.
21. Henning Pagnia and Felix C. Gärtner. On the impossibility of fair exchange without a trusted third party. Technical report, Darmstadt, 1999.
22. Adrian Perrig and Dawn Xiaodong Song. A first step towards the automatic generation of security protocols. In *NDSS*, 2000.
23. A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symp. Found. of Comp. Sci.*, pages 46–57. IEEE Computer Society Press, 1977.
24. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190. ACM Press, 1989.
25. P. Ramadge and W. Wonham. Supervisory control of a class of discrete event processes. *Siam J. Control and Optimization*, 25(1), 1987.
26. P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete-event processes. *SIAM JCO*, 25:206–230, 1987.
27. Hassen Saïdi. Toward automatic synthesis of security protocols. *AAAI Technical Report*, SS-02-05, 2002.
28. Vitaly Shmatikov and John C. Mitchell. Finite-state analysis of two contract signing protocols. *Theor. Comput. Sci.*, 283(2):419–450, 2002.
29. Dawn Song, Sergey Berezin, and Adrian Perrig. Athena: a novel approach to efficient automatic security protocol analysis. *JCS*, 9:2001, 2001.
30. F. Javier Thayer, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(1), 1999.
31. Wolfgang Thomas. Languages, automata, and logic. pages 389–455, 1997.
32. Jianying Zhou, Robert H. Deng, and Feng Bao. Some remarks on a fair exchange protocol. In *Public Key Cryptography*, pages 46–57, 2000.
33. Jianying Zhou and Dieter Gollmann. An efficient non-repudiation protocol. In *PCSFW*, pages 126–132. IEEE Computer Society Press, 1997.
34. Jianying Zhou and Dieter Gollmann. Towards verification of non-repudiation protocols. In *In Proceedings of 1998 International Refinement Workshop and Formal Methods Pacific*, pages 370–380. Springer, 1998.

8 Appendix

Translating protocol models to process models. We now present a translation from the protocol model introduced in Section 2 to the process model introduced in Section 4. We take $\mathit{Moves} = \mathcal{M}$, as the set of process moves, corresponding to the set of all messages in $\mathcal{M}$. For $1 \leq i \leq n$, we map each participant A_{i-1} to a process P_i as follows:

- $X_i = V_{i-1} \cup \{L_i\}$, is the set of variables of process P_i that includes all participant variables V_{i-1} and a special variable L_i corresponding to control points, taking finitely many values in $\mathbb{N}$,
- for all valuations $f \in \mathcal{F}_i[X_i]$, we have $\Gamma_i(f) = \Lambda_{i-1}(f \downarrow V_{i-1})$ and
- $\delta_i : \mathcal{F}_i[\{L_i\}] \times \mathcal{F}_i[X_i \setminus \{L_i\}] \times \mathit{Moves} \mapsto \mathcal{F}_i[\{L_i\}] \times \mathcal{F}_i[X_i \setminus \{L_i\}]$ is the process transition function that exactly corresponds to the participant transition function Λ_{i-1}.

The sets X_i form a partition of $X = \bigcup_{i=1}^{n} X_i$. The set of processes P_i, given all possible behaviors of a fair scheduler Sc, corresponds to the most general exchange program. The realization of a protocol corresponds to a refinement $P'_i \preceq P_i$ for $1 \leq i \leq n$, where each participant A'_{i-1} maps to the process P'_i as follows:

- $X'_i = X_i = V_{i-1} \cup \{L_i\}$ is the set of variables of process P'_i,
- for all valuations $f \in \mathcal{F}'_i[X'_i]$, we have $\Gamma'_i(f) = \Lambda'_{i-1}(f \downarrow V_{i-1})$ and
- for all valuations $f \in \mathcal{F}'_i[X'_i]$, for all moves $m \in \mathit{Moves}$, we have $\delta'_i(f, m) = \Lambda'_{i-1}(f(L_i), f \downarrow V_{i-1}, m)$.

A protocol instance (protocol run) is a trace in $[\![P'_1 \parallel P'_2 \ldots \parallel P'_n \parallel \mathrm{Sc}]\!](v_0)$ for an initial valuation $v_0 \in \mathcal{F}[X]$. The specifications of the participants, which were defined as a set of desired sequences of messages, are subsets of traces in $[\![P'_1 \parallel P'_2 \ldots \parallel P'_n \parallel \mathrm{Sc}]\!](v_0)$. Given specifications φ_i for process P_i, a Y-attack for $Y \subseteq \{P_1, P_2, \ldots, P_n\}$ satisfies φ_i for all $P_i \in Y$, while violating φ_j for at least one process $P'_j \in (\{P_1, P_2, \ldots, P_n\} \setminus Y)'$. There are three participants in a two party fair non-repudiation protocol, the originator O, the recipient R and the trusted third party TTP. We therefore take $n = 3$ in modeling two party fair exchange protocols in the above translation.

We now prove Lemma 2. Given a refinement $P' = (\mathrm{O}', \mathrm{R}', \mathrm{TTP}') \preceq P$, we first characterize the smallest restriction on O' and R' that satisfy the implication conditions:

$$[\![\mathrm{O} \parallel \mathrm{R}' \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{R}}; \text{ and} \tag{7}$$

$$[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}} . \tag{8}$$

We show that for all refinements $\mathrm{R}' \preceq \mathrm{R}$, the implication condition (7) holds. In order to characterize the smallest restrictions on O that satisfies the implication condition (8), we recall the following constraints on O. We show that these constraints are both necessary and sufficient to satisfy (8).

AGS constraints on O. We say that a refinement $\mathrm{O}' \preceq \mathrm{O}$ satisfies the *AGS constraints on O* if O' satisfies the following constraints:

1. $a_1^{\mathrm{O}} \notin \Gamma_{\mathrm{O}'}(v_0)$;
2. $\mathrm{EOO}_k^{\mathrm{O}} \notin \Gamma_{\mathrm{O}'}(\{M_1, \mathrm{EOR}, \mathrm{ABR}^{\mathrm{O}}\})$; and
3. $a_1^{\mathrm{O}} \notin \Gamma_{\mathrm{O}'}(\{M_1, \mathrm{EOR}, M_3\})$.

The most flexible refinements $\mathbf{O}' \preceq \mathbf{O}$ and $\mathbf{R}' \preceq \mathbf{R}$. We now characterize the most flexible refinements $\mathrm{O}' \preceq \mathrm{O}$ and $\mathrm{R}' \preceq \mathrm{R}$ that satisfy the implication conditions $(\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$ and $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{R}}$.

Lemma 4 *For all refinements $R' \preceq R$, the following assertion holds:*

$$[\![O \parallel R' \parallel TTP \parallel Sc]\!] \subseteq (\varphi_O \wedge \varphi_{TTP}) \Rightarrow \varphi_R.$$

Proof Consider an arbitrary refinement $\mathrm{R}' \preceq \mathrm{R}$. We have the following cases of sets of traces of $[\![\mathrm{O} \parallel \mathrm{R}' \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!]$ for the proof:

- *Case 1. Set of traces where m_3 has been received.* For all traces where m_3 has been received, φ_{R} is satisfied. Therefore all these traces satisfy the implication condition, $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{R}}$.
- *Case 2. Set of traces where m_3 has not been received.* For all traces where m_3 has not been received, the traces where either φ_{O} or φ_{TTP} is violated, satisfy the implication condition $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{R}}$ trivially. The interesting case are those traces that satisfy $\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}}$ but violate φ_{R}. These are exactly the traces where O does not have $\mathrm{EOR}_k^{\mathrm{R}}$, since m_4 is not sent before receiving m_3, and R does not have $\mathrm{EOO}_k^{\mathrm{O}}$, as otherwise φ_{R} would be satisfied. We have following cases that lead to a contradiction:
  - *Case (a). O aborts the protocol.* In these traces, since φ_{TTP} is satisfied, the abort token must have been sent to both agents, and since neither agent will be sent the other's signature and the channels between the agents and the TTP are resilient, the traces satisfy φ_{R}, leading to a contradiction.
  - *Case (b). O or R′ resolve the protocol.* In these traces, since φ_{TTP} is true, the TTP sends $\mathrm{EOO}_k^{\mathrm{TTP}}$ to R and $\mathrm{EOR}_k^{\mathrm{TTP}}$ to O and never sends either AO or AR. This implies, given the channel between the agents and the TTP is resilient, the traces satisfy φ_{R}, leading to a contradiction.
  - *Case (c). R′ chooses move ι.* In these traces, since φ_{O} is true, either O aborts the protocol after sending m_1 or she chooses to abort or resolve the protocol after receiving m_2. In either case, given the traces satisfy φ_{TTP}, by the above argument φ_{R} is satisfied as well, irrespective of the behavior of the channel between O and R. This again leads to a contradiction.

Since we have shown that for all traces, either φ_{R} is satisfied or satisfaction of $\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}}$ implies satisfaction of φ_{R}, we conclude that for all refinements $R' \preceq R$ the assertion holds. ■

It follows from Lemma 4, that as R' can always resolve the protocol in state $\{\mathrm{EOO}\}$ and all successor states, such that the resulting trace satisfies $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{R}}$, we have $m_2 \in \Gamma_{\mathrm{R}'}(\{\mathrm{EOO}\})$. Similarly, $m_4 \in \Gamma_{\mathrm{R}'}(\{\mathrm{EOO}, M_2, \mathrm{EOO}_k^{\mathrm{O}}\})$ as φ_{R} is satisfied in all traces where m_3 has been received, thus satisfying $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{R}}$.

In the following lemma, in assertion 1 we show that for all refinements $\mathrm{O}' \preceq \mathrm{O}$ that satisfy the AGS constraints on O, the implication condition (8) is satisfied; in assertion 2 we show that if O' does not satisfy the AGS constraints on O, the implication condition (8) is violated.

Lemma 5 (The smallest restriction on $\mathbf{O}' \preceq \mathbf{O}$) *For all refinements $O' \preceq O$, the following assertions hold:*

1. *if O' satisfies the AGS constraints on O, then*

$$[\![O' \parallel R \parallel TTP \parallel Sc]\!] \subseteq (\varphi_R \wedge \varphi_{TTP}) \Rightarrow \varphi_O.$$

2. *if O' does not satisfy the AGS constraints on O, then*

$$[\![O' \parallel R \parallel TTP \parallel Sc]\!] \not\subseteq (\varphi_R \wedge \varphi_{TTP}) \Rightarrow \varphi_O.$$

Proof Consider an arbitrary refinement $\mathrm{O}' \preceq \mathrm{O}$ that satisfies the AGS constraints on O. We have the following cases of sets of traces of $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!]$ for the proof:

- *Case 1. Set of traces where m_4 has been received.* In the case of classical co-synthesis, an adversarial R will never send m_4 as that satisfies φ_{O} unconditionally, but in assume-guarantee synthesis, from Lemma 4, since all refinements of R satisfy the weaker condition of $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{R}}$, $m_4 \in \Gamma_{\mathrm{R}'}(\langle \mathrm{EOO}, M_2, \mathrm{EOO}_k^{\mathrm{O}} \rangle)$. For all traces where m_4 has been received, φ_{O} is satisfied. Therefore all these traces satisfy the implication condition $(\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$.
- *Case 2. Set of traces where m_4 has not been received.* For all traces where m_4 has not been received, the traces where either φ_{R} or φ_{TTP} is violated, satisfy the implication condition $(\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$ trivially. The interesting case are those traces that satisfy $\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}$ but violate φ_{O}. These are exactly the traces where O does not have $\mathrm{EOR}_k^{\mathrm{R}}$, since m_4 has not been received. We have the following cases that lead to a contradiction:
  - *Case (a). O′ has sent m_3.* In these traces, since O′ satisfies the AGS constraints on O, the only choice of moves for O′ are ι or r_1^{O}; she can wait for R to send m_4 or resolve the protocol. In the set of traces where she eventually receives m_4, by Case 1, the traces satisfy $(\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$. If she does not receive m_4, she will eventually resolve the protocol to satisfy φ_{O}. In the set of traces where she eventually resolves the protocol, since φ_{TTP} is satisfied, and R cannot abort the protocol, the TTP will eventually respond to her request by sending her non-repudiation evidence and not the abort token. These traces therefore satisfy φ_{O}, leading to a contradiction.
  - *Case (b). O′ aborts the protocol before sending m_3.* Since O′ satisfies the AGS constraints on O, she cannot abort the protocol in the initial state v_0. Therefore, O′ must have started the protocol by sending m_1. In all these traces, O′ aborts the protocol after sending m_1 but before sending m_3 and since O′ satisfies the AGS constraints on O, she will not send m_3 after sending the abort request. Since these traces satisfy φ_{TTP}, the abort token must have been sent to both agents, and since neither agent will be sent the other's signature and the channels between the agents and the TTP are resilient, the traces satisfy φ_{O}, leading to a contradiction.
  - *Case (c). O′ resolves the protocol before sending m_3.* In these traces, since φ_{TTP} is true, the TTP sends $\mathrm{EOR}_k^{\mathrm{TTP}}$ to O and $\mathrm{EOO}_k^{\mathrm{TTP}}$ to R and never sends either AO or AR. This implies, given the channel between the agents and the TTP is resilient, the traces satisfy φ_{O}, leading to a contradiction.
  - *Case (d). O′ chooses move ι instead of sending m_3.* In these traces, since φ_{R} is true, R must have resolved the protocol after receiving m_1. In this case, given the traces satisfy φ_{TTP}, by the above argument φ_{O} is satisfied as well. This again leads to a contradiction.
  - *Case (e). The channel between O and R is unreliable.* If either m_1 or m_2 are not delivered, then O′ can abort the protocol. If either m_3 or m_4 are not delivered, then O′ can resolve the protocol. In either case, by Case (a), Case (b) and Case (c), we have φ_{O} is satisfied even when the channel between O and R is unreliable, leading to a contradiction.

We conclude that for all O′ that satisfy the AGS constraints on O, we have $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$.

For assertion 2, consider an arbitrary refinement $\mathrm{O}' \preceq \mathrm{O}$ that does not satisfy the AGS constraints on O. We consider violation of the constraints on a case by case basis. For each case we produce a witness trace that violates the implication condition $(\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$. We proceed as follows:

– *Case 1.* $a_1^O \in \Gamma_{O'}(v_0)$. In a trace where O′ sends an abort request before sending message m_1 in the initial protocol state v_0, it is trivially the case that the trace does not satisfy φ_{O} but satisfies φ_{R}. If the TTP satisfies the AGS constraints on the TTP and sends $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$ in response, then the trace satisfies φ_{TTP}. Therefore, the trace violates $(\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$.
– *Case 2.* $EOO_k^O \in \Gamma_{O'}(M_1, EOO, ABR^O)$. To produce a witness trace we consider a partial trace that ends in protocol state $\{M_1, \mathrm{EOO}, \mathrm{ABR}^{\mathrm{O}}\}$; messages m_1 and m_2 have been received and a_1^{O} has been sent. Since the channel between O and the TTP is resilient, the abort request is eventually processed by the TTP. If O′ sends message m_3 in this state and the TTP responds with move $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$ to her abort request, then there exists a behavior of the channel between O and R such that m_3 is eventually delivered and the protocol is aborted. The trace therefore satisfies $\varphi_R \wedge \varphi_{\mathrm{TTP}}$ but violates φ_{O}; as O cannot get R's signature after the protocol is aborted and R has her signature.
– *Case 3.* $a_1^O \in \Gamma_{O'}(M_1, EOO, M_3)$. To produce a witness trace we consider a partial trace that ends in protocol state $\{M_1, \mathrm{EOO}, M_3\}$; messages m_1 and m_2 have been received and m_3 has been sent. If O′ aborts the protocol in this state and the TTP satisfies the AGS constraints on the TTP and responds with move $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$, then there exists a behavior of the channel between O and R, where m_3 is eventually delivered to R. The trace satisfies $\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}$ but violates φ_{O}.

We conclude that if O′ does not satisfy the AGS constraints on O, then $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!] \not\subseteq (\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$. ∎

From Lemma 5, it is both necessary and sufficient that O satisfies the AGS constraints on O to ensure the implication condition (8).

The maximal refinement $\mathbf{P}^* = (\mathbf{O}^*, \mathbf{R}^*, \mathbf{TTP}^*)$. We recall the definition of the maximal refinement $\mathrm{P}^* = (\mathrm{O}^*, \mathrm{R}^*, \mathrm{TTP}^*)$ below:

1. $\mathrm{O}^* \preceq \mathrm{O}$ satisfies the AGS constraints on O and for all O′ that satisfy the constraints, we have $O' \preceq \mathrm{O}^*$;
2. $\mathrm{R}^* = \mathrm{R}$; and
3. $\mathrm{TTP}^* \preceq \mathrm{TTP}$ satisfies the AGS constraints on the TTP and for all TTP′ that satisfy the constraints, we have $\mathrm{TTP}' \preceq \mathrm{TTP}^*$.

The weak co-synthesis requirement. Let $b \in \mathbb{N}$ be a bound on the number of times that O or the TTP may choose the idle move ι when scheduled by Sc. In the following lemma, for all refinements $P' \preceq \mathrm{P}^*$ that satisfy the AGS constraints on the TTP, in assertion 1 we show that if b is finite, then the condition for weak co-synthesis is satisfied; in assertion 2 we show that if b is unbounded, then the condition for weak co-synthesis is violated.

Lemma 6 (Bounded idle time lemma) *For all refinements $P' = (O', R', TTP') \preceq P^*$ that satisfy the AGS constraints on the TTP, for all $b \in \mathbb{N}$ with O' and TTP' choosing at most b idle moves when scheduled by Sc, the following assertions hold:*

1. *if b is finite, then $[\![O' \parallel R' \parallel TTP' \parallel Sc]\!] \subseteq (\varphi_O \wedge \varphi_R \wedge \varphi_{TTP})$.*
2. *if b is unbounded, then $[\![O' \parallel R' \parallel TTP' \parallel Sc]\!] \not\subseteq (\varphi_O \wedge \varphi_R \wedge \varphi_{TTP})$.*

Proof For the first assertion, we show that the condition for weak co-synthesis holds against all possible behaviors of the channel between O and R. We have the following cases:

– *Case 1. Agents abort or resolve the protocol.* In all traces where the agents abort or resolve the protocol, given b is finite and that TTP′ satisfies the AGS constraints on

the TTP, by Lemma 1 (assertion 1), TTP$'$ will eventually respond to the first and all subsequent requests such that φ_{TTP} is satisfied. In all these traces, given the channels between the agents and the TTP are resilient, both agents get either the abort token or non-repudiation evidences but never both. This ensures φ_{O} and φ_{R} are satisfied.

- *Case 2. The channel between O and R is resilient.* In all traces where neither agent aborts nor resolves the protocol, φ_{TTP} is satisfied trivially. Further, the only refinements of the agents that neither abort nor resolve the protocol are those where $\{m_1, m_3\} \in \mathit{Moves}_{\mathrm{O}'}$ and $\{m_2, m_4\} \in \mathit{Moves}_{\mathrm{R}'}$. Since b is finite, the only choice of moves for O$'$, since she does not abort or resolve the protocol, are m_1 in state v_0 and m_3 in state $\{M_1, \mathrm{EOR}\}$, after choosing at most b idle moves at each state. Similarly, the only choice of moves for R$'$ are ι or m_2 in state $\{\mathrm{EOO}\}$ and ι or m_4 in state $\{\mathrm{EOO}, M_2, \mathrm{EOO}_k^{\mathrm{O}}\}$. If R$'$ never sends m_2, then O$'$ will eventually abort the protocol after bounded idle time and this case reduces to Case 1. If R$'$ never sends m_4, then O$'$ will eventually resolve the protocol after bounded idle time and this case reduces to Case 1. If R$'$ sends m_2 and m_4 eventually, since the channel between O and R is assumed resilient, messages m_1, m_2, m_3 and m_4 are eventually delivered satisfying φ_{O} and φ_{R}.
- *Case 3. The channel between O and R is unreliable.* Since $\mathrm{O}' \preceq \mathrm{O}^*$, we have $a_1^{\mathrm{O}} \notin \Gamma_{\mathrm{O}'}(v_0)$ and $a_1^{\mathrm{O}} \notin \Gamma_{\mathrm{O}'}(\{M_1, \mathrm{EOR}, M_3\})$; O$'$ can abort the protocol in all other states. Therefore, O$'$ satisfies the AGS constraints on O. Since b is finite and O$'$ cannot resolve the protocol before initiating it, the only choice of moves for O$'$ in state v_0 is to send m_1 eventually. If the channel between O and R does not deliver either messages m_1 or m_2, the only choice of moves for O$'$ is to abort the protocol. If either messages m_3 or m_4 are not delivered, then the only choice of moves for O$'$ is to resolve the protocol. In both these cases, since O$'$ chooses to abort or resolve the protocol, by Case 1 the result follows.

We conclude that irrespective of the behavior of the channel between O and R, if b is finite, we have $[\![\mathrm{O}' \parallel \mathrm{R}' \parallel \mathrm{TTP}' \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}})$.

For the second assertion, given an unbounded b, to show that weak co-synthesis fails, it suffices to show that there exists a behavior of the agents, the TTP and the channels that violates the condition for weak co-synthesis. Consider a partial trace ending in protocol state $\{M_1, \mathrm{EOO}, M_2, \mathrm{EOR}, M_3, \mathrm{EOO}_k^{\mathrm{O}}, \mathrm{RES}^{\mathrm{O}}\}$; messages m_1, m_2 and m_3 have been received, R$'$ chooses to go idle, never sending m_4 and O$'$ has sent r_1^{O}. Since b is unbounded, if TTP$'$ chooses to remain idle forever, then φ_{O} and φ_{TTP} are violated leading to a violation of $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}})$. Therefore, given an unbounded b, we have $[\![\mathrm{O}^* \parallel \mathrm{R}^* \parallel \mathrm{TTP}^* \parallel \mathit{Sc}]\!] \not\subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}})$. ∎

From Lemma 6, it is both necessary and sufficient that the refinements $P' \preceq \mathrm{P}^*$ that satisfy the AGS constraints on the TTP, also satisfy bounded idle time to ensure weak co-synthesis. While O and the TTP should satisfy bounded idle time, there are no restrictions on R. Using Lemma 1, Lemma 4, Lemma 5 and Lemma 6 we now present a proof of Lemma 2.

Proof **(Proof of Lemma 2).** In one direction, consider an arbitrary refinement $P' = (\mathrm{O}', \mathrm{R}', \mathrm{TTP}') \in \overline{\mathrm{P}}$. We show that the conditions of assume-guarantee synthesis are satisfied as follows:

- *The implication condition for O.* Since $P' \preceq \mathrm{P}^*$, we have $\mathrm{O}' \preceq \mathrm{O}^*$, $\mathrm{R}' \preceq \mathrm{R}^*$ and $\mathrm{TTP}' \preceq \mathrm{TTP}^*$. As $a_1^{\mathrm{O}} \notin \Gamma_{\mathrm{O}^*}(v_0)$ and $a_1^{\mathrm{O}} \notin \Gamma_{\mathrm{O}^*}(M_1, \mathrm{EOR}, M_3)$, the refinement P' satisfies the AGS constraints on O. Therefore, by Lemma 5 (assertion 1), we have $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$.
- *The implication condition for R.* By Lemma 4, we have $[\![\mathrm{O} \parallel \mathrm{R}' \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{R}}$.

– *The implication condition for the TTP.* Since $\mathrm{TTP}' \preceq \mathrm{TTP}^*$ and TTP' satisfies the AGS constraints on the TTP, by Lemma 1 (assertion 1), φ_{TTP} is satisfied irrespective of the behavior of O and R, which implies $[\![\mathrm{O} \parallel \mathrm{R} \parallel \mathrm{TTP}' \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$.
– *The weak co-synthesis condition.* Given P' satisfies bounded idle time, by Lemma 6 we have $[\![\mathrm{O}' \parallel \mathrm{R}' \parallel \mathrm{TTP}' \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}})$; weak co-synthesis holds.

Since we have shown that the refinement P' satisfies all the implication conditions and the weak co-synthesis condition of assume-guarantee synthesis, we have $P' \in P_{AGS}$. Hence $\overline{\mathrm{P}} \subseteq P_{AGS}$.

In the other direction, consider an arbitrary refinement $P'' = (\mathrm{O}'', \mathrm{R}'', \mathrm{TTP}'') \in P_{AGS}$. We show that $P'' \in \overline{\mathrm{P}}$ as follows:

– *The AGS constraints on O.* By Lemma 5, since it is both necessary and sufficient that a refinement satisfy the AGS constraints on O to ensure the implication condition $(\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$ is satisfied, given the implication condition holds, we conclude that P'' satisfies the AGS constraints on O. Therefore, $\mathrm{O}'' \preceq \mathrm{O}^*$.
– *The AGS constraints on the TTP.* By Lemma 1, since it is both necessary and sufficient that a refinement satisfy the AGS constraints on the TTP to ensure the implication condition $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$ is satisfied, given the implication condition holds, we conclude that P'' satisfies the AGS constraints on the TTP and $\mathrm{TTP}'' \preceq \mathrm{TTP}^*$.
– *The bounded idle time condition.* By Lemma 6, since it is both necessary and sufficient that a refinement satisfy bounded idle time to ensure weak co-synthesis, since weak co-synthesis holds in this case, we conclude that P'' satisfies bounded idle time.
– $P'' \preceq P^*$. Since we have shown that $\mathrm{O}'' \preceq \mathrm{O}^*$ and $\mathrm{TTP}'' \preceq \mathrm{TTP}^*$, we have $P'' \preceq \mathrm{P}^*$.
– $P_* \preceq P''$. Since P_* is the smallest refinement in the set P_{AGS}, given $P'' \in P_{AGS}$, it must be the case that $\mathrm{P}_* \preceq P''$.

For $P'' \in P_{AGS}$, as we have shown that $\mathrm{P}_* \preceq P'' \preceq \mathrm{P}^*$, P'' satisfies the AGS constraints on the TTP and satisfies bounded idle time. Thus we have $P'' \in \overline{\mathrm{P}}$ and hence $P_{AGS} \subseteq \overline{\mathrm{P}}$. The result follows. ∎

We now present a proof of Lemma 3. We recall the *enhanced AGS constraints on the TTP* below:

1. *Abort constraint.* If the first request received by the TTP is a_1^{O}, then her response to that request should be $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$; If the first request received by the TTP is a_1^{R}, then her response to that request should be req^{O};
2. *Resolve constraint.* If the first request received by the TTP is a resolve request, then her response to that request should be $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$; If the TTP receives res^{O} in response to req^{O} within bounded idle time, then her response should be $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$, otherwise it should be $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$.
3. *Accountability constraint.* If the first response from the TTP is $[x, y]$ or the first response from the TTP is req^{O} and the next response is $[x, y]$, then for all subsequent abort or resolve requests her response should be in the set $\{\iota, x, y, [x, y]\}$.

Proof **(Proof of Lemma 3).** From Protocol 2, since the refinement O_s does not abort the protocol either in the initial state v_0 or after sending message m_3, we have O_s satisfies the AGS constraints on O. By our definition of the behavior of TTP_s, we have TTP_s satisfies the enhanced AGS constraints on the TTP. From the definition of the main protocol in Protocol 2 and the abort subprotocol in Protocol 3, since the resolve subprotocol is identical to the KM protocol, we have O_s and TTP_s satisfy the bounded idle time requirement. We take $A = \{\mathrm{O}, \mathrm{R}, \mathrm{TTP}\}$ and show that there is no Y-attack for $Y \subseteq \{\mathrm{O}, \mathrm{R}\}$ through the following cases:

- *Case 1.* $|Y| = 2$. In this case $Y = \{\mathrm{O}, \mathrm{R}\}$. We show that $[\![\mathrm{O} \parallel \mathrm{R} \parallel \mathrm{TTP}_s \parallel \mathrm{Sc}]\!] \subseteq \varphi_{\mathrm{TTP}}$. For all traces in $[\![\mathrm{O} \parallel \mathrm{R} \parallel \mathrm{TTP}_s \parallel \mathrm{Sc}]\!]$ where R does not abort the protocol, since TTP_s satisfies the enhanced AGS constraints on the TTP, by Lemma 1 (assertion 1), φ_{TTP} is satisfied. For all traces where R sends an abort request, the TTP sends req^{O}. If O responds with res^{O} within bounded idle time, then the TTP resolves the protocol for both O and R such that the AGS constraints on the TTP are satisfied. If O does not send res^{O} within bounded idle time, then the TTP aborts the protocol, such that the AGS constraints on the TTP are satisfied. For all subsequent abort requests from R, the TTP response satisfies the AGS constraints on the TTP. All traces therefore satisfy φ_{TTP}. Hence, there is no Y-attack in this case.
- *Case 2.* $|Y| = 1$. In this case, either $Y = \{\mathrm{O}\}$ or $Y = \{\mathrm{R}\}$. We have the following cases towards the proof:
  - *Case (a).* $Y = \{O\}$. We show that $[\![\mathrm{O} \parallel \mathrm{R}_s \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{R}}$; it will follow that $[\![\mathrm{O} \parallel \mathrm{R}_s \parallel \mathrm{TTP}_s \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{R}}$. Consider the set of traces in $[\![\mathrm{O} \parallel \mathrm{R}_s \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!]$. For all traces where R does not abort the protocol, by Lemma 4, we have $\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}} \Rightarrow \varphi_{\mathrm{R}}$. For all traces where R aborts the protocol, if he has received m_3, then φ_{R} is satisfied. For all traces where R aborts the protocol and message m_3 has not been received, if φ_{TTP} is violated, then the implication holds and if φ_{TTP} is satisfied, then either both agents get abort tokens or their respective non-repudiation evidences, thus satisfying φ_{R}. We have shown that all traces satisfy the implication condition $\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}} \Rightarrow \varphi_{\mathrm{R}}$. Since we have a fixed TTP that satisfies the AGS constraints on the TTP, we have φ_{TTP} is satisfied in all traces by Case 1. As φ_{O} is satisfied by assumption, we conclude φ_{R} is satisfied as well. Therefore, there is no Y-attack in this case.
  - *Case (b).* $Y = \{R\}$. It can be shown that $[\![\mathrm{O}_s \parallel \mathrm{R} \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!] \not\subseteq (\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$. We show that, by fixing the TTP, we have $[\![\mathrm{O}_s \parallel \mathrm{R} \parallel \mathrm{TTP}_s \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$. Consider the set of traces $[\![\mathrm{O}_s \parallel \mathrm{R} \parallel \mathrm{TTP}_s \parallel \mathrm{Sc}]\!]$. For all traces where R does not abort the protocol, since O satisfies the AGS constraints on O, by Lemma 5, we have $(\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$. If R aborts the protocol, since the TTP satisfies the enhanced AGS constraints on the TTP, and the channel between O and the TTP is operational, req^{O} must have been received by O. At this stage, if O_s has sent message m_3, then the only choice of moves for O_s to satisfy φ_{O} is res^{O}; a request to resolve the protocol. Since the channels are operational, there exists a bound on the idle time of the TTP such that both req^{O} and res^{O} can be delivered within this bound. Moreover, as TTP_s satisfies the enhanced AGS constraints on the TTP, both O and R will be issued non-repudiation evidences and never abort tokens, thus satisfying φ_{O}. If O_s has not sent message m_3, then the only choice of moves for O_s to satisfy φ_{O} are ι or res^{O}. In all these traces, since TTP_s satisfies bounded idle time and the AGS constraints on the TTP, either both agents get non-repudiation evidences or abort tokens but never both, thus satisfying φ_{O}. Therefore, all these traces satisfy $(\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$, which given φ_{R} is satisfied by assumption and φ_{TTP} is satisfied by Case 1, implies φ_{O} is satisfied as well. There is no Y-attack in this case.
- *Case 2.* $|Y| = 0$. In this case $Y = \emptyset$ and $(A \setminus Y)' = \{\mathrm{O}_s, \mathrm{R}_s, \mathrm{TTP}_s\}$. Since P_s satisfies bounded idle time, in all traces where R does not abort the protocol, by Lemma 6, the condition for weak co-synthesis is satisfied. In all traces where R aborts the protocol, as TTP_s satisfies the AGS constraints on the TTP, she sends req^{O}. In all these traces, since TTP_s and O_s satisfy bounded idle time, and the channels are operational, O_s chooses ι or sends res^{O} and TTP_s responds with either abort tokens or non-repudiation evidences

but not both, leading to the satisfaction of φ_{O} and φ_{R}. Since φ_{TTP} is satisfied by Case 1, all these traces satisfy $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}})$. Therefore, there is no Y-attack in this case.

The result follows. ■

Proof **(Proof of Theorem 8).** By Lemma 3, it follows that if the TTP does not change her behavior, then P_s is attack-free. Further, by the weak co-synthesis condition, we have $[\![\mathrm{O}_s \parallel \mathrm{R}_s \parallel \mathrm{TTP}_s \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}})$ and hence by Theorem 1, we have $[\![\mathrm{O}_s \parallel \mathrm{R}_s \parallel \mathrm{TTP}_s \parallel \mathrm{Sc}]\!] \subseteq \varphi_f$. Thus P_s satisfies fairness. Using PCS we ensure abuse-freeness. Since $[\![\mathrm{O}_s \parallel \mathrm{R}_s \parallel \mathrm{TTP}_s \parallel \mathrm{Sc}]\!] \cap (\diamond\mathrm{NRO} \wedge \diamond\mathrm{NRR}) \neq \emptyset$, the refinement P_s enables an exchange of signatures and hence is an exchange protocol. We conclude that if the TTP does not change her behavior, then P_s is an attack-free fair non-repudiation protocol. ■